Commodore
COMPUTER CLUB
51
L. 4.500
La rivista degli utenti di sistemi Commodore
25 marzo 1988 - Anno VII - N. 51
Sped. Abb. Post. Gr. III/70
CR - Distr. Mepe
Speciale L.M. per C 128
Tu come Spielberg
Un programma per fare i cartoon
La vita segreta di Amiga
C 16
Tutte le poke che devi sapere
BIT PARADE
RRARI FORMULA ONE
Systems
CAMPUS
Il tuo software-lab questo mese a 48 pagine

IN EDICOLA

N. 3 - Lire 12.000
Commodore 64 Club
ZAGOR
Dischetto
a due facce
oltre 300 Kbyte
di software
• ZAGOR
• RISICOM 64
• PALE MOON
• PERSIAN GULF
• CITY KILLER
• MAGIC VIDEO
systems

Sommario

RUBRICHE

In copertina: Fotogramma del film "Fievel sbarca in America" di Steven Spielberg

Direttore: Alessandro de Simone - **Caporedattore:** Michele Maggi
Redazione/collaboratori: Paolo Agostini, Davide Ardizzone, Claudio Baiocchi, Luigi Callegari, Sergio Camici, Umberto Colapicchioni, Maurizio Dell'Abate, Valerio Ferri, Roberto Ferro, Cristina Magnaghi, Giancarlo Mariani, Roberto Marigo, Clizio Merli, Marco Mietta, Marco Miotti, Oscar Moccia, Roberto Morassi, Guido Pagani, Antonio Pastorelli, Sonja Scharrer, Fabio Sorgato, Valentino Spataro, Danilo Toma
Segretaria di redazione: Maura Ceccaroli **Grafica:** Arturo Ciaglia, Gabriella Galbusera
Direzione, redazione, pubblicità: v.le Famagosta, 75 - 20142 Milano - Tel. 02/8467348
Pubblicità: Milano: Leandro Nencioni (direttore vendite), Guido Agosti, Giorgio Ruffoni, Claudio Tidone - v.le Famagosta, 75 - 20142 Milano - Tel. 02/8467348
• Emilia Romagna: Spazio E - P.zza Roosevelt, 4 - 40123 Bologna - Tel. 051/236979
• Toscana, Marche, Umbria: Mercurio srl - via Rodari, 9 - San Giovanni Valdarno (Ar) - Tel. 055/947444
• Lazio, Campania: Spazio Nuovo - via P. Foscari, 70 - 00139 Roma - Tel. 06/8109679
Segreteria: Paola Bertolotti - **Abbonamenti:** Liliana Spina
Tariffe: prezzo per copia L. 4.500. Abbonamento annuo (11 fascicoli) L. 45.000. Estero: il doppio. Abbonamento cumulativo alle riviste Computer e Commodore Computer Club L. 85.000.
I versamenti vanno indirizzati a: Systems Editoriale Srl mediante assegno bancario o utilizzando il c/c postale n. 37952207
Composizione: Systems Editoriale Srl - **Fotolito:** Systems Editoriale Srl
Stampa: Systems Editoriale/La Litografica Srl - Busto Arsizio (Va)
Registrazioni: Tribunale di Milano n. 370 del 2/10/82 - Direttore Responsabile: Michele Di Pisa
Sped. in abb. post. gr. III - Pubblicità inferiore al 70% - **Distrib.:** MePe - via G. Carcano, 32 - Milano
Periodi Systems: Banca Oggi - Commodore Club (disco) - Commodore Computer Club - Commodore Computer Club (disco produzione tedesca) - Computer - Computer disco - Electronic Mass Media Age - Energy Manager - Hospital Management - MondoRicambi - Nursing '90 - PC Programm (disco) - Personal Computer - Security - Software Club (cassetta ed. italiana) - VR Videoregistrare

PIONIERI, MA IN RITARDO

Recentemente la stampa non specializzata ha dato molto risalto ad un "rivoluzionario" centro servizi informatico; un po' di sarcasmo, comunque, non guasta...

Dunque, anche in Italia la meccanizzazione (termine vetusto che risale ai tempi delle schede perforate) ha inizio; anzi, sembra che proceda lungo il cammino intrapreso dai tempi, appunto, dei calcolatori a valvole.

Il ministro di turno promette "un collegamento tra numerose banche dati in modo che sia possibile, in tempo reale, chiedere, ad un solo sportello, più certificati contemporaneamente".

Per ora la straordinaria innovazione sarebbe limitata al comune di Roma, in cui confluirebbero tutti i dati del Ministero della Difesa, dell'Inps, del comune di Roma e dell'Enpas (a proposito, quest'ultimo non era stao abolito?).

Il condizionale è comunque d'obbligo perchè, a quanto pare, il fantascientifico progetto, per esser realizzato, necessita di locali in cui sistemare le apparecchiature.

Dunque, una rivoluzione che parte ancora con il piede sbagliato, dopo oltre vent'anni dalle prime proposte in questo senso.

La notizia riportata con grande enfasi (ne ha parlato perfino il TG), in un'epoca come la nostra, suscita amarezza per almeno due motivi: il primo è che il sistema informatico cui ci si riferisce sembrerebbe nato al servizio della elefantiaca ed anacronistica Burocrazia, che assume ancor più la forma di Istituzione della nostra Repubblica; in secondo luogo perchè non ci si vergogna neanche un po' di presentare come innovativo un sistema, quello informatico appunto, già in vigore da parecchi anni in molti paesi industrializzati.

Del resto non ci si può aspettare di più da una classe politica che, in piena era informatica, impedisce, di fatto, lo svilupparsi della posta elettronica, unico, fra i vari rimedi, che tenderebbe ad annullare ben noti ritardi e disservizi.

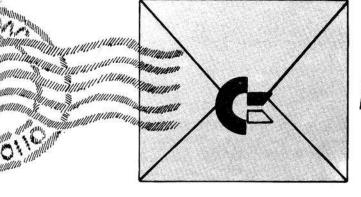
Ancora oggi, 1988 (quasi vent'anni dopo lo sbarco sulla Luna) assistiamo, grazie alla Burocrazia, a code che gli automobilisti sono costretti a fare per pagare ridicole integrazioni del bollo auto: il burocrate d'ordinanza non ha dimostrato sufficiente fantasia pensando di far pagare la tassa con il prossimo bollo.

Ma quello del bollo-auto, come forse il lettore avrà, suo malgrado, sperimentato, non è che un esempio minimo e, sostanzialmente, trascurabile in confronto alla vera inefficienza con cui vengono diretti numerosi enti pubblici.

Non sno affatto rari, come alcune immagini televisive ci mostrano spesso, gli esempi di pratiche (spesso inutili) portate avanti a mano, non da monaci amanuensi, ma da impiegati dello stato.

Nel Far-West i pericoli erano tanti, ed i pionieri cercavano di evitarli.

Ma la stessa condotta non viene seguita dai nostri burocrati, specie animale (per nulla in estinzione), che sembrerebbe, per giunta, protetta da leggi speciali.

Alessandro de Simone

la vostra posta

Ha da accendere?

☐ A volte il led rosso del mio drive, che accendo dopo aver dato tensione al computer, rimane illuminato per lungo tempo. E' normale questo fenomeno?
(Luca Vignale - Brandizzo)

• Di solito un computer, o una periferica, "lancia" un segnale di inizializzazione non appena viene "svegliato" dall'invio della corrente elettrica.

Nel 99% dei casi è del tutto indifferente accendere prima il computer, e poi le periferiche, o viceversa.

Tuttavia nei vari manuali, e non solo relativi ad apparecchi Commodore, si consiglia sempre, per evitare inconvenienti di vario tipo, di accendere per ultimo, tra i vari apparecchi, proprio il computer.

Non dobbiamo infatti dimenticare che è questo il vero direttore d'orchestra dell'allegra compagnia: il segnale di reset che giunge dal calcolatore, al momento della sua accensione, è più... perentorio di quelli auto-generati dalle varie periferiche che, per una serie di motivi, possono non essere avvertiti dalla periferica stessa.

Periodicità di Commodore

☐ Con quale frequenza pubblicate la rivista "Commodore" di cui mi sono procurato due soli fascicoli nell'arco di un anno?
(Massimo B. Taranto)

• "Commodore" era una rivista dalla periodicità mensile che, per una serie di motivi, fu soppressa in seguito alla decisione di riportare, su Commodore Computer Club, gli argomenti di maggior interesse.

La testata, tuttavia, non è stata liquidata del tutto ed è riservata alla pubblicazione di argomenti monografici di un certo interesse.

Il primo, dedicato al linguaggio macchina ed alle routine grafiche di Toma, fu pubblicato, appunto, circa un anno fa.

Il secondo, monografico sul Totocalcio, ha visto la luce verso la fine dell'87.

Che cosa ci riserverà il terzo fascicolo?...

Spiegazioni incomplete

☐ Protesto perchè nell'articolo "A scuola di Raster" (C.C.C. n. 47) non avete spiegato perchè, modificando il passo 16, il video vibra cambiando colore in modo incontrollabile, cosa che non succede con passo 8 e 16.
(Sergio Manferdini - S.Lazzaro)

• Il programma "Raster su sfondo multicolore", cui si riferisce il nostro lettore, consente di ottenere l'effetto di colorare ogni coppia di righe di schermo con un colore diverso.

Il "passo" di cui si parla è il secondo valore valore Data (16) presente in riga 1008.

Modificando questo valore con 8 (oppure con 4) si ottiene, rispettivamente, la modifica del colore in ciascuna riga e ogni... mezza riga!

Valori diversi impediscono un'alternanza "modulare" dei colori che, quindi, cambiano ad ogni schermata riproducendo il fastidioso tremolio lamentato.

Andare in profondità, per scoprire il "passo" a causa del quale si perde la modularità, è sicuramente un'operazione interessante che, tuttavia, richiede del tempo.

Il programmino presentato, comunque, aveva solo lo scopo di sti-

Interfaccia Midi

☐ E' possibile collegare l'interfaccia Midi della Siel con la tastiera Technics SX-k700?
(Anonimo timido)

• L'interfaccia Midi consente di controllare strumenti musicali elettronici (tipicamente: tastiere, drums e così via) per mezzo di un personal computer, dotato di analoga interfaccia, grazie allo scambio vicendevole di dati.

Un'interfaccia universale (quale è la Midi) ha proprio questo di bello: la possibilità di collegare tra loro più strumenti in grado di inviare, e ricevere, informazioni secondo uno standard ben definito.

La semplice dichiarazione "Midi compatibile" dovrebbe quindi esser sufficiente per garantire il collegamento tra un C/64 ed una qualsiasi tastiera musicale.

In pratica, però, ti consiglio di rivolgerti ad un negozio specializzato che sia in grado di assicurare non solo il corretto funzionamento, ma soprattutto la disponibilità del software adeguato senza il quale, non dimenticarlo, gli strumenti resterebbero "separati", benchè collegati via cavo tra loro.

molare l'interesse del lettore che, per proprio conto, dovrebbe portare avanti da solo certi suggerimenti.

Molto spesso, come si può riscontrare, ospitiamo volentieri considerazioni di lettori che, spinti da curiosità, risolvono problemi posti, in vari modi, anche da queste stesse colonne.

Mini curiosità

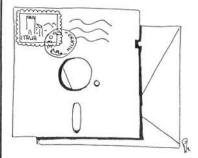
□ Vi invio due microlistati per C/64 che, a mio parere, sono simpaticissimi.
(Gianluca Ghioni - Roma)

• Il primo programmino altera il normale ciclo di interruzione interno e può essere utilizzato, sugerisce Gianluca, come un contasecondi.

Prima che l'elaborazione termini, tenete premuto il tasto Run/Stop finchè non compare nuovamente il cursore: sarà interessante notare il modo in cui il computer gestisce lo scrolling del video.

```
1 poke 56325,0
2 for a=1 to 60
3 print a: next
4 poke 56325,58
```

Il secondo listato è invece banalissimo dal momento che si limita a cambiare in rapida successione i colori del bordo e del fondo. Lo pubblico, comunque, ad uso e consumo dei super-ultra-mega-extra-principiantissimi (ci sono anche loro, diamine!)

```
1 for a=1 to 15
2 poke 53280,a
3 poke 53281,a
4 next: goto 1
```

Drive 1541 e Amiga

□ E' possibile collegare il drive 1541 (o compatibile C/64) all'Amiga?
(Gianluca Marini - Bologna)
(Fausto Ancelini - S.G. Persiceto)

• Nel campo dell'informatica tutto è possibile; naturalmente lo affermiamo da un punto di vista strettamente tecnico.

Tra il dire e il fare, però, c'è di mezzo l'oceano del protocollo di comunicazione, della possibilità di sincronizzare comandi che operano a velocità e simbolismi diversi, della questione dei connettori e, non ultima, la reale utilità pratica di un simile collegamento.

Spieghiamoci meglio: se l'utente possiede solo l'Amiga (dotato, di per sè, di un drive interno) e desidera aggiungere un altro drive, non ha alcun bisogno di arrovellarsi il cervello dal momento che può acquistare direttamente un disk drive, specifico per Amiga, che spesso costa meno di un 1541 da "riciclare".

Il problema, ragionevolmente, sorge soltanto per l'ex-utente di C/64, passato all'Amiga, che vorrebbe sfruttare, se possibile, le periferiche di cui già dispone; nel caso specifico, drive interno (di Amiga) e drive esterno (1541 e simili).

Ma a quale scopo? Non certo per riversare su 1541 i dati elaborati dai programmi professionali di Amiga che richiedono due drive: il 1541 non verrebbe "riconosciuto", a meno che non si voglia cambiare la Rom ed altre minuterie con una spesa di certo superiore ad una buona carrettata di drive "veri" per Amiga.

L'unico modo di utilizzarlo sarebbe quello di riversare (secondo uno standard più realizzabile, pur se con gran fatica) dati e programmi di creazione dell'utente stesso. Ma in questo caso è proprio necessario usare due drive contemporaneamente, o non è meglio risparmiar fatica (e pazienza) limitandosi ad un solo drive?

Si potrebbe obiettare che, in commercio, esiste già un package hard-soft in grado di far girare, su Amiga, per mezzo di un 1541, i programmi che giravano su C/64.

Ma ecco subito pronta un'altra obiezione: se un utente possiede un 1541, vuol dire (nel 99.99% dei casi) che possiede anche un C/64. A che pro, dunque, far girare su Amiga i programmi che possono girare tranquillamente (e, soprattutto, senza errori) sul C/64 originale?

E se uno (ma quante ipotesi!) non posside un C/64?

Bè, in questo caso abbiamo a che fare con veri e propri maniaci: chi mai, infatti, possedendo l'Amiga (ed i meravigliosi giochi di cui dotarlo) si sognerebbe di utilizzarlo al 10% accontentandosi di far girare programmi, decisamente modesti, sviluppati per computer di classe inferiore?

Paragoni

□ E' possibile fare un paragone tra il chip sonoro del C/64 e quello dell'Amiga?
(Gabriele Piroddi - Cagliari)

• I due circuiti sono spaventosamente diversi e la capacità stereofonica offerta dall'Amiga è solo una delle caratteristiche che li pone su piani

Speed Dos per drive compatibili

□ E' possibile applicare lo Speed Dos ai drive 1541 compatibili?
(Da alcune telefonate)

• Lo Speed Dos ha avuto un notevole successo presso gli utenti dei drive 1541 dal momento che permette notevoli aumenti di velocità nella trasmissione dei dati da e verso la periferica.

Purtroppo, a causa delle diversità delle circuiterie dei drive compatibili, non è possibile montarvi lo Speed Dos (di cui esistono, tra l'altro, numerose versioni).

A tale inconveniente sembra aver provveduto la ditta Newel di Milano che ha risolto il problema proponendo una Rom, da inserire al posto di quella presente sulla piastra del C/64, lasciando quindi libera la porta di espansione per altri utilizzi.

L'economica Rom (chiamata "Star-Dos"), oltre ad offrire la maggior velocità di trasferimento dati e programmi, contiene numerosi altri comandi aggiuntivi, decisamente comodi per chi lavora spesso con il drive.

nettamente diversi tra loro.

Non si sa ancora molto del circuito integrato Paula (nome dato al chip di Amiga), ma quel poco che è noto fa prevedere incredibili potenzialità, tutte da scoprire.

Uno dei paragoni che si può fare con gran facilità è quello che vede impegnati i due chip, SID e Paula, alle prese con la riproduzione della voce umana.

Nel caso del C/64 si devono fare non pochi salti mortali per raggiungere un livello di comprensibilità accettabile; con l'Amiga, invece, vi sono addirittura specifici comandi, anche da Basic, per far parlare il computer.

Senza tener conto che siamo solo all'inizio: quando l'Amiga verrà sfruttato a fondo ne... sentiremo davvero delle belle.

Ignazio Soddu (di Fluminimaggiore), suggerisce un espediente per incrementare le potenzialità della stampante Mps-801:

"Non tutti i modelli Mps-801 presentano il tasto per l'avanzamento della carta, benchè sia effettivamente presente sul circuito della stampante.

Per attivarlo, quindi, è sufficiente staccare il coperchio sotto il quale, posizionato alla sinistra della spia di accensione, è presente un rettangolino di gomma.

Incollate un pezzetto di foglio di alluminio (del tipo usato per conservare i cibi) delle stesse dimensioni. Una volta rimesso a posto il coperchio, premetelo in corrispondenza del rettangolino per veder avanzare la carta.

Fast visibile

□ E' possibile attivare, con un C/128, la procedura FAST senza "spegnere" lo schermo?
(Antonio Luongo - Pienza)

• Un computer, durante il suo funzionamento, è costretto ad effettuare numerose elaborazioni delle quali non ci rendiamo conto dal momento che la procedura è totalmente automatica, cioè non richiesta espressamente dall'utente e, come si dice in gergo, "trasparente" al normale funzionamento dell'elaboratore.

Tra le varie routine che il calcolatore gestisce mi limiterò a ricordare quella che, istante per istante, controlla se non sia stato premuto il tasto Run/Stop, in modo da interrompere subito la procedura in corso.

Oltre a questa appena accennata è anche presente una routine che verifica la presenza del pennello elettronico del monitor, e che quindi è responsabile della corretta visualizzazione dei caratteri presenti sullo schermo, della varietà dei colori e così via.

Nonostante tutte queste routine agiscano alla velocità del linguaggio macchina, richiedono, come puoi intuire, un certo tempo che, apparentemente modesto per una sola operazione, diventa ragguardevole se consideriamo il fatto che ognuna di queste routine viene attivata ogni 60mo di secondo (in un minuto, quindi, una routine in interrupt viene "chiamata" ben 3600 volte!).

Se, però, riusciamo ad individuare momenti in cui alcune routine possono esser considerate superflue, possiamo risparmiar tempo semplicemente impedendo la loro esecuzione.

Nel caso specifico del comando Fast, quindi, è necessario rinunciare alle routine più lente, tra le quali figura quella che sovrintende il video.

La "scomparsa" dello schermo, quindi, è una condizione necessaria per favorire l'aumento della velocità operativa.

Colpa dei puntatori

□ Ho provato a digitare il programma "Carica Koala" (C.C.C. n.47) che, dopo alcuni insuccessi, a volte funziona e a volte no. Da che cosa può dipendere?
(Giacomo Spinelli - Siena)

• Quando si digitano programmi che alterano puntatori vitali del Ba-

Il Vic 20 è morto?

□ Del Vic 20 non si sente più parlare. Chi lo possiede ancora, deve proprio buttarlo via?
(Da alcune lettere)

• Non direi; in fin dei conti è pur sempre un computer al quale tutti noi dobbiamo qualcosa.

E' certamente vero il fatto che non vale la pena rivenderlo come usato: pur se perfettamente funzionante, il prezzo di mercato della configurazione base non può superare le 70 mila lire (buone solo per una romantica cenetta per due in un discreto ristorante).

Semmai il Vic 20, unito ad un vecchio televisore che nessuno usa più in famiglia, può diventare una valida stazione di invio e ricezione dati via modem, un utile terminale su cui elaborare programmi in Basic, una palestra hardware su cui compiere le operazioni più pericolose che non compromettano l'esistenza del più prezioso C/64 o Amiga, posizionati, poco più in là, sullo stesso tavolo.

Non è poi vero che nessuno si occupa di commercializzare accessori per il glorioso computer: tra gli altri (pur se non numerosi) la ditta Niwa, di Sesto San Giovanni, dispone ancora di numerosi accessori (espansioni, cartucce, paddle) che propone a prezzi, ovviamente, proporzionali al reale valore del sistema.

sic (inizio e fine programma, zona variabili, puntatori Load e Save e così via) bisogna tassativamente registrare il programma PRIMA di lanciarlo.

In caso di insuccesso è indispensabile spegnere il computer, riaccenderlo, caricare la versione precedentemente registrata, studiarne le possibili cause, apportarvi le correzioni e, dopo averla nuovamente registrata, dare un secondo Run; e così via.

Se, infatti, un programma mal digitato lo si fa partire con troppa disinvoltura, dopo la segnalazione di Syntax Error (o altre che, comunque, interrompono il programma) è molto probabile che i puntatori, ormai alterati, confondano le idee al computer che non sa più in che zona si trovi il listato Basic (oppure lo ritiene ubicato in tutt'altra parte).

In alcuni casi il computer si impianta del tutto ma, molto più spesso di quanto non si creda, continua a sopravvivere, pur se boccheggiando come una bestia in fin di vita.

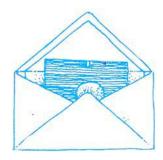
Ne consegue che, in una situazione così anomala, un programma può funzionare perfettamente(!). Ricaricato, però, in condizioni "normali" (dopo, cioè, che è stato spento e riacceso), eventuali file registrati quando il calcolatore era in condizioni anomale non vengon più riconosciuti, oppure allocati chissà dove, procurando gli inconvenienti lamentati.

Telefonare prima dell'uso

□ **Non sarebbe più semplice, per noi lettori e per voi della redazione, evitare di telefonarvi prima di spedire programmi per l'eventuale pubblicazione?**
(Maurizio C. - Salerno)

• Alcuni anni fa, a causa della carenza di programmatori esperti, invitavamo i lettori, semplicemente, ad inviare listati ed articoli.

Oggi il numero di utenti in grado di scrivere un buon programma è cresciuto di molto e, tra questi, sono tantissimi coloro che, desiderosi di veder pubblicato un lavoro, spendono un sacco di soldi per inviarlo a mezzo raccomandata, a volte espresso, se non, addirittura, assicurata.

Per evitare delusioni (un rifiuto, si sa, è sempre doloroso) abbiamo deciso di invitare i lettori a parlarne prima per telefono, in modo da sapere subito se il programma proposto può avere speranze di esser pubblicato oppure no.

Naturalmente esaminiamo anche la corrispondenza di coloro che non seguono la procedura descritta; ma è intuitivo che, per questioni di correttezza, esaminiamo dapprima i dischetti di coloro con i quali abbiamo raggiunto accordi preventivi per via telefonica.

Capisco benissimo che per vari motivi, tra cui la timidezza, alcuni lettori provano una certa soggezione nell'uso del telefono.

Naturalmente la timidezza non si vince con un fiume di parole; da parte nostra, però, non possiamo fare altro che ripetere che, tra i nostri compiti, c'è anche quello di rispondere al telefono ai nostri lettori, tra i quali sono presenti anche ragazzi giovanissimi che, a torto, ritengono di non dover "disturbare" i componenti della redazione di un giornale.

Bando alla timidezza, quindi, e telefonateci, meglio se al pomeriggio: se non altro, per evitare di spender soldi nell'invio di plichi postali che rischiano di restare senza risposta.

Hardware

Due consigli per MPS

Il nostro lettore Corrado Simoni (di Mestre) suggerisce un metodo per ovviare alla scarsa reperibilità del nastro per la stampante Mps-803:

"Acquistate una cartuccia per Olivetti Lexicon 80,82,83 DL (8X5 mm.) che, tra l'altro, costa circa un terzo di quella Commodore originale.

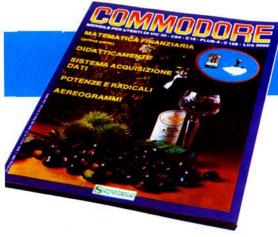
Apritela e srotolate il nastro facendo ben attenzione a non formare spirali. Sovrapponete le estremità del nastro, per una lunghezza di circa 5 cm., su un foglio di giornale immobilizzandole lateralmente (in modo che non sfuggano).

Sui due spezzoni sovrapposti appoggiate un foglio di carta e scaldate al fuoco la lama di un coltello la cui larghezza sia almeno pari a quella del nastro. Appoggiate quindi la lama, calda ma non troppo, sulla carta, in corrispondenza delle due estremità del nastro in modo che il calore possa trasmettersi uniformemente; esercitate una certa pressione affinchè i due spezzoni, una volta fusi a causa del calore, si saldino tra loro in seguito al raffreddamento.

La parte più difficile è quella relativa all'inserimento del nastro nella vecchia cartuccia per Mps-803.

Dopo alcuni tentativi, comunque, l'operazione ha successo."

128 KBYTES

SINCLAIR COM

COMMODORE

MSX

DI RIVISTA.

PUTER

Personal computer

STUDIO DI FUNZIONE

RILOCATORE DI PROGRAMMI

FUNZIONE VAL PER IL QL

TRE RIVISTE IN UNA!

E' IN EDICOLA

Personal computer è la rivista Systems per gli utenti Commodore, MSX, Sinclair. Non solo tre riviste per tre diversi utenti: **Personal Computer** è anche un'idea nuova per far comunicare tutti gli hobbisti. **Personal Computer:** 128 Kbytes di rivista, tutti i mesi in edicola. L'abbonamento cumulativo a **Computer** e **Personal Computer** costa solo L. 65.000.

Il mercato si evolve. Anche noi.

Non solo per C/128

□ Volendo, in seguito, passare ad un sistema professionale, sono intenzionato a comprare una stampante ed un monitor da utilizzare, per ora, con il mio C/128.
(Remo Bibbiani - Vada)

• Vi sono molte stampanti che, prodotte di serie per computer professionali (dotati di interfaccia Rs-232, Centronics, IEEE-488) dispongono, tuttavia, di interfacce specifiche per il C/64 (valide, quindi, anche per C/16 e C/128) con cui risultano completamente compatibili.

Una stampante che non possiede l'etichetta "Commodore compatibile" può, tuttavia, esser collegata (sempre mediante opportuna interfaccia) ad un C/64, ma non potrà mai funzionare con i programmi grafici specifici per C/64, come Print Master, Print Shop e TUTTI quelli che consentono di riversare, su carta, schermate in alta risoluzione.

E' ovvio che ti conviene procurarti fin da ora una periferica "Commodore compatibile" che, grazie al semplice, rapido (ed economico) scambio di cartucce, potrai usare, in seguito, anche con altri computer (Ms-Dos, Amiga ed altri ancora).

Il monitor 1901, cui accenni nella lettera, lo posseggo io stesso e lo uso, di solito, sia con il C/128 (in modo 40 e 80 colonne) sia con il mio AT/ compatibile, senza effettuare la minima modifica.

Anzi, durante il trasferimento dei dati (da C/64 ad AT), spesso utilizzo il monitor 1901 come sistema di visualizzazione unico per entrambi i computer: con il C/128 in modo 64 (40 colonne) e con l'AT (connettore RGB).

Naturalmente, per controllare ciò che succede sui due computer, agisco spostando il comodo deviatore posto sul frontalino del 1901.

Da CP/M a CP/M

□ Posseggo numerosi dischetti, in formato CP/M Mfm, contenenti programmi vari e vorrei caricarli sul mio C/128 dotato di 1571. Come posso fare?
(Luigi Adorni - Cinisello)

• Purtroppo non posso fare altro che ribadire che il CP/M è un sistema operativo morto e sepolto; sarà quindi molto difficile sfruttare le potenzialità del microprocessore Z-80 presente sulla piastra dei nostri C/128.

Nessuno, ormai, si interessa più di CP/M e lo dimostra il fatto che spesso, da queste stesse pagine, abbiamo rivolto appelli ai lettori che, in un modo o in un altro, facciano, con il CP/M, qualcosa di più che leggere la directory del dischetto presente nella confezione del computer.

Sproteggere Gw-Basic

□ E' possibile, con una cartuccia sprotettrice, registrare schermate grafiche eventualmente visualizzate, ed il Tool stesso oltre che il programma scritto in Gw-Basic?
(Antonello Jannone - Cormano)

• Molte cartucce, di cui abbiamo spesso parlato, consentono di riversare su nastro (o disco) l'intero contenuto della memoria del calcolatore mediante la pressione di un tasto presente sulla stessa cartuccia.

In questo modo, pertanto, si aggira l'ostacolo della protezione applicata a quei programmi che impediscono operazioni di duplicazione.

E' molto probabile che anche con il nostro Gw-Basic tali cartucce compiano la loro azione; ma toglimi una curiosità: a che serve ricorrere all'uso di simili espedienti? I nostri programmi non sono per nulla protetti e li puoi trasferire con la massima semplicità utilizzando normalissimi copiatori; per non parlare dell'opzione di formattazione che trasferisce automaticamente, sul nuovo dischetto, il sistema operativo Ms-Dos.

Uno strano numero

□ Formando, da qualsiasi postazione telefonica, il numero di telefono 00.98.21.32.04 è possibile ascoltare la propria voce...
(Smanettone anonimo di Amiga e modem)

• "Il numero di telefono -afferma il nostro lettore- l'ho trovato grazie ad un programma ricevuto, via modem, da alcuni miei amici statunitensi; questi, a loro volta, lo hanno ottenuto come elaborazione di un programma incaricato di rintracciare un numero di telefono che non corrispondesse, sull'intero pianeta, a nessun abbonato."

Se la spiegazione, ben poco razionale, sia una balla o meno (il numero mi è stato comunicato in pieno pe-

Aiutateci a servirvi meglio

• Spesso alcuni lettori, che dichiarano di possedere numeri arretrati della nostra rivista, porgono quesiti le cui risposte sono già state esplicitamente pubblicate (in occasione di risposte ad analoghe domande) oppure sono contenute in articoli presenti nei fascicoli in loro possesso.

Per evitare di ripetere argomenti già trattati, pertanto, ricordate di indicare sempre, nelle lettere che ci inviate, i numeri dei fascicoli in vostro possesso: potremmo infatti indicarvi, se esistono, gli articoli che, in un modo o in un altro, possono chiarire gli argomenti richiesti.

riodo carnevalesco), non sono in grado di saperlo.

Certo è che funziona!

Ho provato, con il massimo scetticismo, a formulare il numero ed a ripetere, come un deficiente, "Pronto, pronto" finchè non ho sentito la mia stessa voce sfalsata di quasi un secondo.

Tra l'altro il contatore SIP sembra che non scatti (posseggo un contascatti ed ho potuto verificarlo) e la "procedura" funziona anche con un telefono pubblico che, al termine della "conversazione" restituisce il gettone.

Quale sarà il mistero legato al bizzarro numero di telefono?

L'anonimo smanettone assicura che la voce di colui che telefona effettua il giro del mondo, servendosi, addirittura, di satelliti di telecomunicazione, per poi ritornare alla base, sfalsata di qualche decimo di secondo.

SED e CLD

☐ Che funzione hanno le istruzioni l.m. SED e CLD?

(Federico Mestrone - Volvera)

• L'istruzione SED (SEt Decimal mode) impone al microprocesore di "ragionare", da quel momento in poi, non più in modo esadecimale, ma in decimale.

Una delle conseguenze più vistose è che, nell'effettuare somme, il bit di riporto (carry) non verrà più generato quando si supera il valore 15 ($0F) ma quando si supera 9, proprio come siamo abituati nel sistema decimale.

Puoi immaginare facilmente, quindi, che cosa succede se il microprocessore elabora, in modo decimale, routine scritte per funzionare in modo esadecimale!

Per tornare al modo "standard" è presente l'istruzione CLD (Clear Decimal Mode) che, appunto, cancellando il modo decimale, riattiva, automaticamente, il modo esadecimale.

Perchè il drive

□ Da un po' di tempo tutte le riviste di informatica, tra cui la vostra, sembrano aver dimenticato coloro che posseggono il solo registratore a cassette...
(Da alcune lamentele)

• L'hobby dell'informatica ha superato da tempo la fase pionieristica in cui l'unica memoria di massa sufficientemente economica era rappresentata dall'unità a nastro.

Tutti noi ci siamo limitati a sognare per qualche annetto il disk drive a causa del suo prezzo relativamente elevato (quasi due milioni di lire, nel lontano 1980). Al giorno d'oggi, grazie al progresso della tecnologia, sono stati drasticamente ridotti sia i prezzi dei drive originali Commodore, sia quelli dei "compatibili" (leggi: spudorata copia hardware di quelli Commodore).

Oggi come oggi, quindi, non sembra giustificato l'illusorio risparmio ottenibile dal mancato acquisto del drive.

Guardandoci attorno notiamo che altri hobby, di vario tipo, richiedono cifre "minime" decisamente superiori rispetto a quelle richieste da un personal computer: gli appassionati di sci, tanto per fare un esempio, pagano fior di bigliettoni per soddisfare la propria voglia di sport (avete idea di quanto costi una settimana bianca, attrezzatura a parte?); chi preferisce l'aeromodellismo ha da spendere una bella cifra tra aerei, motori, radiocomandi; per non parlare, poi, degli appassionati di motociclette o di hi-fi.

Qualunque hobby, insomma, richiede una cifra non indifferente per consentire di trarre un reale giovamento.

E non credo di essere smentito affermando che un personal computer (ed in particolare un C/64) è il passatempo di gran lunga più economico, soprattutto se confrontato con altri diversivi "intelligenti".

La nostra particolare attenzione nei confronti del drive, quindi, non solo nasce da una precisa richiesta dei lettori, ma soprattutto dalla consapevolezza che, con un drive, si possono ottenere soddisfazioni per nulla paragonabili a quelle offerte da un semplice datassette.

E chi ha fatto il gran passo sa bene ciò che affermo.

Emulatore Ms-Dos per C/64

• Numerosi lettori, già acquirenti della prima versione Ms-Dos per C/64, si lamentano del fatto di non aver avuto la possibilità di usufruire dello sconto speciale di cui si parlava nella prima confezione.

Teniamo a precisare che, al momento del "lancio" di Ms-Dos V.1, non immaginavamo di ottenere lo strepitoso successo che, invece, ha riscosso il package di cui parliamo.

Il talloncino di sconto, contenuto nella prima confezione, era da considerare come un incoraggiamento a procurarsi la versione successiva che, secondo le nostre previsioni, avrebbe avuto una diffusione minore (solo per appassionati) e, di conseguenza, un prezzo di copertina certamente superiore.

Considerato, però, il volume di vendita realizzato con la prima realise, non solo abbiamo deciso di non incrementare il prezzo (nonostante la duplicazione, su disco, sia più costosa di quella su nastro) ma, addirittura, di diminuirlo a L.19000 generalizzando, di fatto, lo sconto promesso.

Chi ha acquistato la prima versione a L.25000, e considerando quanto detto in queste note, non potrà fare a meno di apprezzare, ce lo auguriamo, la nostra presa di posizione; questa, infatti, risulta del tutto contraria a quella che inviterebbe ad "approfittare" del successo di un prodotto, partendo dal presupposto che, con ogni probabilità, si venderebbe egualmente a qualunque prezzo.

risposte rapide

Programmi copiatori
(Bruno Del Frate - Milano)
(Corrado Simoni - Mestre)
Non penso che sarà possibile pubblicare programmi di utilità che consentano di effettuare copie di file; di solito questi programmi l.m. sono lunghi diverse migliaia di byte e sarebbe stressante, per i lettori, digitarli.

Hardware made in Italy
(Franceso Carlino)
Come puoi notare, da un po' di tempo vengono presentati vari progetti di piccoli accessori per computer.

Programmi professionali
(Giovanni Giulietti - S.Pietro)
Il nostro consiglio è quello di rivolgersi alla software house che distribuisce il prodotto specifico.

C/64 che scotta
(Andrea Renzi - Roma)

E' molto probabile che il computer disponga di una scarsa ventilazione perchè "soffocato" da libri o altri oggetti posizionati nei pressi. Prova a sollevarlo dal tavolo mediante spessori di alcuni centimetri, tenendo lontane lampade o altre fonti di calore.

Nessun errore
(Giacomo Rizzo - Avellino)
La riga del listato cui ti riferisci non esiste nemmeno sul programma originale. L'errore, quindi, è da attribuirsi ad una errata digitazione del programma.

Più informazioni
(Matteo Ferrero - Alba)
Non esiste un libro che parli specificamente delle protezioni da applicare al software per C/64.

Puoi esaminare, però, i numerosi articoli che compaiono, piuttosto spesso, sulla nostra rivista. Tieni presente, però, che quasi sempre è necessario sapere qualcosa di linguaggio macchina.

Renumber L.M.
(Fabrizio Aliprandi - Seregno)
E' pressochè impossibile realizzare un programma utility in grado di rilocare automaticamente un programma scritto in l.m., a causa di motivi più volti descritti su queste stesse pagine.

L'unico modo per rilocare un programma l.m. è quello di scriverne uno servendosi di assemblatori (ottimo è il Macro Assembler della Commodore) che, memorizzati sotto forma di file Assember, oltre che l.m., permettono di esser "ripresi" e modificati in qualsiasi momento.

Non implementato
(Michele Galante)
Un comando non implementato è un comando la cui parola codice è presente nel set di istruzioni della macchina, ma che non ha alcun effetto pratico perchè, appunto, in corrispondenza dell'indirizzo cui si riferisce non è presente alcuna routine.

Si tratta, in definitiva, di comandi che i progettisti avrebbero voluto inserire nel linguaggio commercializzato (di solito il Basic) ma che poi, per una serie di motivi, sono stati costretti ad eliminare.

Per non esser costretti a riallocare tutte le parole codice, si preferisce, a questo punto, lasciare la parola codice al suo posto, ma privarla di un qualsiasi effetto.

Soluzioni
(Marco Brugnanca - Lavis)

L'idea di pubblicare, a distanza di tempo, le soluzioni delle varie adventure è interessante e vedremo di svilupparla adeguatamente.

1541-II
(Paolo Monticelli - Codogno)
Il 1541-II è il nuovo modello di drive che prenderà il posto del 1541. Nulla a che vedere, quindi, con le potenzialità del 1571, purtroppo uscito di produzione.

Schermate grafiche
(Fulvio Gioa - Prà)

Spiegare come fare per caricare più schermate grafiche, da far apparire parzialmente insieme ad una parte della pagina testo, non è un'operazione semplicissima.

Ti assicuro, comunque, che stiamo per affrontare l'argomento Hi-Res in modo sistematico; nel frattempo impegnati con il l.m. e studia a fondo tutti i programmi di grafica che abbiamo pubblicato finora.

Scacchi da memorizzare
(Nicola Cardaci - Pisa)

Se un programma commercializzato non possiede, tra le altre, l'opzione di registrazione di partita non completata, purtroppo non c'è nulla da fare e sarebbe un'impresa disperata tentare di alterare il programma (di solito scritto in l.m.) per inserire l'opzione mancante.

Per un appassionato giocatore di scacchi, comunque, non penso che sia sufficiente il Plus/4. Sembra che l'Amiga disponga di programmi di scacchi che batterebbero anche Gorbaciov (oppure Kasparov o Karpov, non ricordo bene).

Bei disegni
(Paolo Biguzzi - Cesena)

I grafici che hai inviato sono molto belli. Per ottenerli ad una maggiore velocità penso che sia necessario compilare il programma che li ha generati.

Hai mai pensato, però, di utilizzare il "Compilatore grafico-matematico"? Chiedilo al nostro servizio arretrati: il programma è stato sviluppato appositamente per realizzare grafici matematici tridimensionali ad alta velocità. ("Commodore Club", su nastro, N.1).

LA VITA SEGRETA DI AMIGA

Alcuni pettegolezzi sulla nascita di Amiga e sull'insolita gestione dei file su disco

di **Luigi Callegari**

Se è vero che a scuola si studia la storia della letteratura e della filosofia, per comprenderle meglio, crediamo che conoscere quali siano le radici ed i "genitori" di Amiga possa meglio illuminarci su alcune caratteristiche speciali ed esclusive del suo sistema operativo.

Pochi sanno che quando Lorraine (allora Amiga si chiamava così!) cominciò a trasformarsi, da schemi elettronici, ad hardware funzionante (nel lontano 1984) ci furono molti problemi per adattarle un sistema operativo che supportasse caratteristiche hardware talmente avanzate ed innovative.

I suoi creatori volevano un sistema operativo che agisse in Multitasking, in grado di gestire efficientemente i circuiti VLSI Custom (Agnes, Paula e Denise) ideati da Jay Miner, in grado di supportare I/O sincroni ed asincroni, hardware indipendente e tanto altro ancora.

La prima ditta incaricata di creare un tale S.O. fallì nel proprio lavoro, non essendo riuscita a produrre in tempo utile, per la presentazione ufficiale della macchina, qualcosa di funzionante. Venne allora interpellata una piccola software house inglese, la Metacomco, composta da soli venticinque tecnici programmatori molto esperti.

Essa disponeva già di alcuni linguaggi scritti in BCPL (linguaggio precursore del C) e, soprattutto, il sistema operativo Tripos, anch'esso scritto in BCPL. Venne dunque commissionato l'adattamento ad Amiga di questo sistema operativo, sviluppato all'università di Cambridge nel 1976 originariamente per sistemi DEC PDP-11 e Sage.

Grazie alle caratteristiche di Tripos, già molto vicine a quelle richieste dai produttori di Amiga, ed al fat-

to che era scritto in BCPL (linguaggio estremamente "portatile") la conversione in AmigaDOS avvenne in sole tre settimane, producendo complessivamente circa 192K di codice assembly 68000!

In realtà, per migliorarne l'efficienza, il Kernel (nocciolo "primordiale" del sistema) fu scritto in puro assembler, facilitando anche il lavoro della Metacomco, in quanto l'interfaccia software con gli specialissimi chip "custom" (copper e blitter, particolarmente) era quindi già stata fornita dagli ideatori di Amiga.

Finalmente, nel febbraio 1985, il gruppo di tecnici che aveva costruito Amiga, investendo più di due anni di duro lavoro ed una trentina di milioni di dollari (non di loro tasca, come preciserà più tardi Jack Tramiel), vedeva il proprio figlioletto di silicio funzionare da solo, col suo cuore software, invece di essere tenuto in vita, come era stato fatto sino ad allora, da un minicomputer Sun che simulava il sistema operativo.

Lo staff di programmatori produsse ex-novo anche Intuition, il sistema a finestre ed icone peculiare di Amiga, mentre dobbiamo ringraziare i consigli della Metacomco se venne creato anche CLI, per gli utenti che preferiscono lavorare con i tasti invece che col mouse.

Originariamente il Basic che doveva accompagnare la macchina avrebbe dovuto essere prodotto dalla Microsoft (la leggendaria software house americana, resa famosa dalle produzioni per Apple), ma tardò ad effettuare i necessari ampliamenti alla versione 5.2 del linguaggio (quella del Macintosh).

Infatti si desiderava un Basic in grado di sfruttare appieno le caratteristiche dei circuiti VLSI grafici e sonori esclusivi di Amiga, memori delle penose limitazioni del Basic del C/64, ove per fare qualunque cosa si doveva ricorrere a PEEK e POKE. Ancora venne in aiuto la Metacomco, che produsse, o meglio convertì partendo da testi in BCPL, i linguaggi Pascal, MacroAssembler, Lisp, C ed ABasic appositamente per Amiga, in tempo utile per la presentazione ufficiale della macchina negli USA.

Il Basic fornito oggi con Amiga anche in Italia è chiamato AmigaBasic, invece di ABasic, ed è stato sviluppato dalla Microsoft. Si pensi che questo primitivo linguaggio aveva ancora i numeri di linea, mancava delle procedure con parametri e di comandi come SAY, TRANSLATE e del FILL che sfrutta direttamente il blitter!

Tripos e Amigados

L'area di maggiori differenze tra Tripos, quindi AmigaDOS, e gli altri S.O., risiede nella gestione concorrente dei file.

Tripos è basato sul concetto di "task" (compito) e trasferimenti di messaggi multipli. Quando viene eseguito un task applicativo (in pratica, un programma) esso trova, contemporaneamente, molti altri task già in azione.

In particolare, esiste sempre un task per ogni "device" (periferica) disponibile al sistema, sebbene la maggior parte di essi sarà in stato di riposo, perchè attivati solo alla bisogna.

Poichè vi è un solo processore in Amiga, un task colloquia con gli altri con messaggi inviati tramite apposite porte di comunicazione. Dal momento che il processore 68000 lavora in divisione di tempo tra i vari task, si troverà, dunque, ad inviare un messaggio da un task (in pratica a inviarlo in un particolare buffer di RAM) ed a rileggerlo quando eseguirà il task a cui era stato inviato (che quindi rileggerà automaticamente da quel buffer di RAM).

In questo modo ogni task "ritiene" di avere a sua completa ed unica disposizione sia un processore 68000 che altri task con i quali scambiare dati; ogni task, per conto suo, gode della stessa "illusione" grazie alla gestione delle "porte" software di intercomunicazione.

Ciò spiega, tra l'altro, perchè, quando si esegue una scrittura su disco, l'accesso effettivo a questo (accensione del led) avviene poi con ritardi di tempi apparentemente casuali, insoliti per i noti sistemi CP/M e MS-DOS.

L'interfaccia di trasferimento di dati è molto simile a quella di UNIX ed è identica per ogni periferica ed applicazione. Ciò consente al programmatore una grande linearità nella stesura di programmi in assembler e, ancora più, in C.

Struttura del file Amigados

Nel campo della struttura dei file, Tripos-Amigados è essenzialmente diverso da qualunque altro sistema operativo realizzato per un personal computer.

Tanto per iniziare, non vi è una

traccia di directory, ovvero una tabella dei nomi di file presenti su disco; esiste invece, al suo posto, un "root block" (traccia radice) posta al centro del disco che contiene il nome del volume, con data ed ora di formattazione, seguiti da una "hash table".

Per i non-ingegneri e non-informatici diremo che la tecnica di "hashing" ("spezzatino", letteralmente) consente di trasformare un nome di file (caratteri ASCII) direttamente in un numero di blocco di memorizzazione sul disco; ciò è possibile usando, appunto, una tabella detta "hash table" ed alcune sofisticate manipolazioni matematiche.

Questo sistema è usato regolarmente in sistemi dotati di grandi memorie di massa e permette, tra le altre cose, un accesso molto più rapido quando sono presenti numerosi file.

Il blocco ricavato dallo hashing del nome contiene, a sua volta, una directory od un file che controlla una struttura di directory "ad albero" come in UNIX e PC-DOS 2 (quella con directory che contengono file ed altre sub-directory, a loro volta contenenti altri file e sub-directory, ed avanti all'infinito).

Senza approfondire troppo i dettagli tecnici, tralasciando di spiegare, tra l'altro, come viene risolta la collisione di codici hash (i file "pippo" e "maria" potrebbero produrre lo stesso codice hash!), ci limiteremo a dire che, con questa tecnica, il sistema è in grado di gestire file di lunghezza virtualmente infinita con grande velocità, specie quando si richiede l'ampliamento degli stessi per inserire nuovi dati.

Inoltre, tutti i file sono ad accesso casuale, non vi sono distinzioni tra file binari, ASCII od altro agli occhi del DOS, nè si ravvisa la necessità di inserire fisicamente speciali codici di controllo (tipo EOF, end of file, per individuare la fine di un file). Tutti i file sono cosiderati democraticamente eguali tra loro con conseguente linearità di gestione.

Altra particolarità degna di nota è che ogni blocco di dati singolo "conosce" il file cui appartiene e contiene i necessari puntatori sia al blocco precedente che a quello successivo.

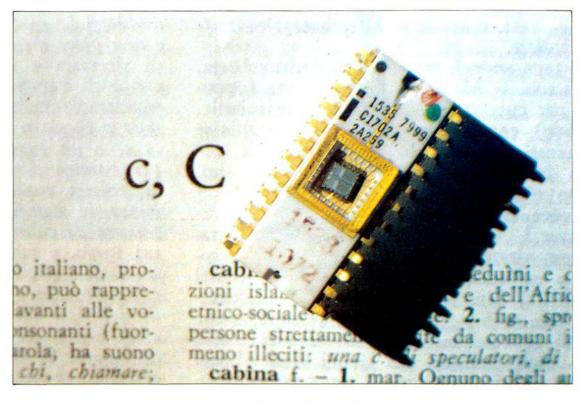
Ne consegue che perfino nei casi in cui il disco viene danneggiato, oppure viene "perduta" l'intestazione di un file, vi sono ottime probabilità di recuperare l'intero contenuto del disco; mentre, infatti, in PC-DOS un minimo danno alla traccia di direttorio può condurre l'utente sull'orlo del suicidio, con AmigaDOS basta usare DISKDOCTOR (presente sul dischetto di Workbench) od analoghi programmi per recuperare automaticamente il contenuto del disco.

L'unico svantaggio di questo sistema consiste nel fatto che la directory, o la ridenominazione di files, è molto più lenta che in qualunque altro sistema.

Altra caratteristica di Tripos è la sua capacità di conoscere contemporaneamente ed in tempo (quasi) reale i nomi dei volumi presenti nei floppy (o hard disk) collegati: accede direttamente alla periferica necessaria senza che l'utente debba dichiarare esplicitamente su quale numero di periferica scrivere o leggere ed è in grado di richiedere l'inserimento (in un qualunque drive) del dischetto necessario se questo, per errore, risulta assente nella sede dovuta.

Concludiamo riportando una curiosità: molti si chiederanno perchè il led del drive resta acceso per alcuni secondi nonostante il computer abbia già terminato l'accesso (letto o scritto il file, fatto partire il programma e così via).

Nel breve lasso di tempo il sistema attiva un particolare task interno (chiamato "disk-validator") per memorizzare sul dischetto la nuova mappa dei blocchi occupati sul disco ("bitmap"), necessaria ad AmigaDOS; se si sfila il dischetto prima dell'esecuzione di questa operazione, la "Bitmap" resta marcata come "invalida" e quando si reinserirà il dischetto il sistema presenterà la classica finestrella (in alto a sinistra) con la scritta "Disk is not validated" mentre attiva il task di ricostruzione della Bitmap e di convalida del dischetto.

Quante cose si muovono da sole sotto le nostre dita senza che ce ne accorgiamo, vero?

SERENA VARIABILE

Una tranquilla chiacchierata istruttiva sulla definizione delle variabili nel linguaggio compilatore "C"

di **Luigi Callegari**

Tipi di dati in C

In C non esistono tipi di dati molto astratti come, ad esempio, in Modula2 od in Pascal (insiemi o tipi definiti dal programmatore stesso); esistono, invece, vari tipi "base" e la possibilità di crearne nuovi unendo tra loro i tipi basilari già a disposizione.

Inoltre il C offre un tipo di variabile molto particolare, il "puntatore", peculiarità dell'Assembler, che permette la scrittura di codici molto flessibili e portatili. Si ricordi che un buon programmatore C deve sempre sfruttare al massimo i puntatori. D'altro canto, però, i "pointers" sono anche tra i maggiori ostacoli all'apprendimento del linguaggio.

Ogni compilatore, in dipendenza anche dall'elaboratore su cui opera, prevede diversi tipi di dati (come estensione dei valori accettabili, precisione matematica, occupazione di memoria per ciascuna variabile); è necessario dunque consultare il manuale d'uso fornito col programma per avere un'idea precisa di come lavora la versione di C in vostro possesso.

SCHEDA TECNICA

Articolo didattico specifico per il linguaggio "C"

Hardware richiesto: qualsiasi computer munito di compilatore C

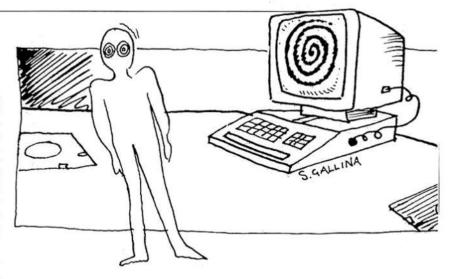
Si suggerisce la lettura del fascicolo n.50

Consigliato, tuttavia, il Computer Amiga dotato di compilatore Lattice C

Consigliato agli esperti.

Ciò che garantisce lo standard C è il fatto che, secondo i dettami di Kernighan & Ritchie, un dato di tipo intero (int) non è mai più lungo di un dato di tipo corto (short) e che esso è sempre minore od uguale al numero di bytes occupati da un dato di tipo lungo (long). Inoltre, nella quasi totalità dei casi, un dato di tipo "carattere" occupa un solo byte, ma ciò non è pragmatico. In linguaggio C, vedrete, c'è ben poco di pragmatico...

Tutti gli "oggetti" del C possono essere mescolati con grande libertà, poichè non viene effettuato, dal compilatore, un controllo molto stretto sui tipi di dati e si considera praticamente sempre possibile convertire i tipi tra di loro.

Ad esempio, assegnando ad una variabile definita come "per numeri interi" (int) una costante di carattere (ad esempio, "c"), operazione che farebbe arrabbiare qualunque compilatore Basic, Pascal, Fortran, Ada o Modula2, per il C significherebbe solo assegnare il valore ASCII (tipicamente, se non si usa lo standard EBCDIC od altri nel caso si lavori con un mini-computer) numerico di quel carattere alla variabile numerica.

Parimenti, assegnando ad una variabile intera il contenuto di una variabile in singola od in doppia precisione, significa, per il C, effettuarne automaticamente la riduzione e l'arrotondamento per cercare di farla stare nell'angusto spazio della variabile intera.

Questo concetto di "forzatura" o "casting" dei dati è un altro dei cardini del figlioletto informatico di Ritchie.

Tutto ciò lascia al bravo programmatore una grande libertà espressiva, mentre toglie al dilettante la protezione "materna" del compilatore da possibili errori concettuali, tipica di tutti gli altri linguaggi compilati. Inoltre, gli errori divengono assai più difficili da rilevare nel testo, specie se lungo.

Una nota esplicita meritano le "stringhe", ovvero gli insiemi di caratteri. Questo tipo di dato non esiste in C, ovvero il linguaggio non è ufficialmente in grado di trattare direttamente parole o frasi come dati di tipo semplice. Ci si deve ricordare molto bene di ciò, perchè quando si incomincia a parlare di matrici (array) di caratteri e dei corrispondenti puntatori (che sono i mezzi con cui in C si gestiscono le stringhe alfanumeriche) chi è abituato, con il Basic, a trattare una variabile di caratteri quasi come una variabile numerica, si confonderà facilmente le idee.

Le variabili in "C"

In parole molto povere, un programma è una lista di istruzioni, scritte secondo le regole formali di un linguaggio, in grado di accettare dati in ingresso, elaborarli sfruttando i circuiti (hardware) dell'elaboratore e produrre, finalmente, risultati.

Il concetto esposto è generale ed applicato ovunque: per "ingresso di dati" si può intendere il digitare centinaia di coordinate spaziali di stelle oppure lo smanettare furiosamente la manopola del joystick, per ottenere, come "risultati finali", rispettivamente, un modello matematico di costellazione, oppure il movimento videogiocoso di PacMan.

Il cuore di ogni linguaggio è quindi costituito da "dati", che è in grado di maneggiare. Ogni linguaggio è in grado di maneggiarne tipi variamente disparati, ma tipicamente, tra questi, vi sono almeno i numeri interi (senza parte frazionaria) compresi tra -32768 e +32767 e caratteri (lettere).

Chi, ad esempio, conosce un moderno interprete Basic (come AmigaBasic) saprà che le seguenti variabili:

a%   b&   c!   d#   e$

possono rappresentare cinque diversi tipi di dati, ovvero: numero intero "corto", numero intero "lungo", numero in singola precisione, numero in doppia precisione e stringa di caratteri.

Poichè i dati vengono manipolati tramite variabili, descrivendo i tipi di variabili esistenti in un linguaggio si definiscono anche gran parte dei dati gestibili.

In questo articolo ci occuperemo, principalmente, della definizione delle variabili nel linguaggio "C".

C e aggiornamenti

Il compilatore Lattice C della Metacomco è giunto, ormai, alla versione 4.0 ed è ora ospitato su ben quattro dischetti. La versione più completa costa intorno ai trecento dollari, ma circa settecentomila lire presso alcuni import... profittatori italiani.

Il compilatore "Aztec C" della Manx è giunto alla versione 3.4 e costa circa 350 dollari.

Ambedue sono ottime implementazioni, complete delle librerie per sfruttare le funzioni evolute del S.O. di Amiga.

Riportiamo gli indirizzi delle Software House americane:

Metacomco plc
26 Portland Sq.
Bristol England BS2 8RZ

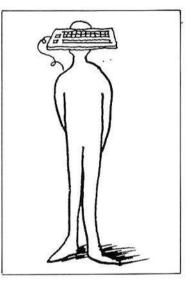
Manx Software Systems
One Industrial Way East
Eatontown, NJ 10541
USA

Le dichiarazioni

Molti dei linguaggi compilati richiedono la dichiarazione del tipo di ognuna delle variabili che si intende usare. Ciò significa che il programmatore, all'inizio del testo sorgente, deve specificare che cosa simboleggi ognuno dei nomi di variabili che userà nel corso del programma.

Tralasciando di trattare l'enorme teoria logica e formale che sta dietro ad ogni compilatore di linguaggio (tanto affascinante quanto complessa e vasta), vorremmo almeno spiegare che la regola della dichiarazione esplicita dei tipi di variabili permette, nel caso di C, almenc tre vantaggi:

- costringere il programmatore a predefinire bene che cosa voglia fare.
- permettere al compilatore di generare un codice ottimizzato (ovvero, ogni variabile occupa sempre la minima memoria necessaria).
- consentire di accorgersi e segnalare (WARNING) eventuali operazioni di "forzatura" sui tipi di dati, a volte consentite, ma talvolta (spesso...) frutto di errori del programmatore.

Come sappiamo, la dichiarazione delle variabili è obbligatoria in C. Il tipo della variabile è espresso (quasi) interamente dallo specificatore di tipo, e non dai caratteri post-fissi, come in Basic ("!" per intero e "$" per le stringhe).

Eccezione notevole sono i puntatori, che vengono indicati da un asterisco prima del nome. Esempi:

```
int jacky;
int *julia;
```

In questo caso "jacky" è dichiarata come variabile intera, "julia" come puntatore a numero intero.

Le parole specificatrici di tipi basilari in C sono le seguenti sette:

char, short, int, long, unsigned, float, double

Le parole "long", "short" ed "unsigned" possono intendersi come aggettivi, in quanto accompagnano solitamente "int" o "float" per meglio definirli. Sono dunque ammesse anche le dichiarazioni:

short int, long int, unsigned int, long float

Si noti che "essere accettate" non significa che il compilatore generi necessariamente variabili differenti. Infatti, ad esempio, "long float" equivale a "double". Non sono tipicamente ammessi costrutti del tipo "unsigned float" o "short float", ma è bene consultare il manuale della implementazione in vostro possesso.

Ciò significa che il compilatore co-

TIPO	BIT	CAMPO
char	8	-128
unsigned char	16	-32768...+32767
short	16	-32768...+32767
unsigned short	16	0...+65535
int	32	+/- 2147483648
unsigned int	32	0...+4294967295
long	32	+/- 4294967295
float	32	+/- 10e-37...+/-10e38 (6,7 cifre di precisione)
double	64	+/- 10e-37...+/-10e-38 (15,16 cifre di precisione)
(pointer)	32	(indirizzo assoluto)

munque non segnala errore, ma assume le variabili come se fossero state dichiarate senza "unsigned" o "short".
Altre equivalenze tipiche sono le seguenti:

short è identico a short int
long è identico a long int
unsigned è identico a unsigned int
unsigned short è identico a unsigned short int

Teoricamente (ciò che avviene in pratica dipende dal compilatore) la specifica "unsigned" indica che il numero è sempre positivo, "long" che si richiede una maggiore precisione (e quindi un maggiore numero di bytes per l'allocazione della variabile), "short" il contrario di "long" ovvero minore precisione e meno bytes.

Ovviamente, è fondamentale dichiarare appropriatamente le variabili. Pensate, infatti, all'evidente caso in cui si voglia il risultato di un calcolo scientifico in virgola mobile, impossibile da incamerare in una variabile intera; più sottile, ma altrettanto sciocco, è il ricorso ad una variabile in virgola mobile come contatrice di un ciclo che opera solo con numeri interi.

Questo non tanto per lo spreco di bytes, necessari ad immagazzinare il contenuto della variabile, ma soprattutto perchè la variabile viene gestita assai meno velocemente, dal momento che l'elaboratore effettua i calcoli in virgola mobile con efficienza molto minore del caso in cui usa una variabile intera.

Ricordiamo che, come abbiamo detto in C.C.C. n.50, il microprocessore lavora sempre con numeri interi e per ottenere la matematica in virgola mobile deve fare un gran numero di operazioni elementari, anche per una semplice somma.

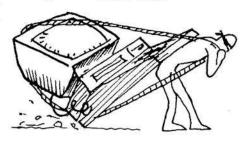
I bravi programmatori in AmigaBASIC sanno che un ciclo del tipo...

FOR i=1 to 10000: ..... :NEXT i

...è sicuramente più lento di un ciclo...

FOR i%=1 to 10000: ....: NEXT i

...in quanto il secondo usa una variabile intera.

Ogni implementazione di C ha le proprie limitazioni circa i nomi delle variabili, perciò bisogna consultare il manuale. Tipicamente il numero di caratteri ammessi e considerati è sovrabbondante e, per i nostri scopi, è praticamente inutile sapere quali siano gli esatti valori di limite.

Diremo che un compilatore, come "Lattice" per Amiga, conserva tutti i caratteri digitati e tutti sono considerati significativi, anche perchè il limite della lunghezza di linea, fissata a 256 caratteri, fornisce già una limitazione intrinseca.

Ricordiamo che un carattere di un nome di variabile si dice "significativo" quando viene effettivamente considerato dal compilatore. Si pensi, ad esempio, che certe vecchie versioni di Microsoft Basic (vedi C/64) consideravano significativi solo i primi due caratteri, perciò anche se il programmatore usava i nomi "genoveffa" e "gerolamo" il linguaggio considerava sempre la stessa variabile, in quanto i primi due caratteri ("ge") sono identici e gli altri ignorati. Non è così invece per la maggior parte dei compilatori

I tipi base del Lattice C

Elenchiamo i tipi base del Lattice C, il compilatore "ufficiale" di Amiga, riportando, per ciascuno, nome, numero di bit richiesti in memoria per l'immagazzinamento e campo di estensione.

Per chi si voglia divertire a compilare "qualcosa" e vedere un esempio pratico di uso di variabili di tipo diverso in C, proponiamo un banale programmino che calcola l'età dell'utente. Si intuirà, in tal modo, che anche per un programma tanto modesto sono necessarie molte informazioni di cui parleremo a lungo prossimamente.

```
/* Programma esemplificativo di calcolo dell' eta'
   utilizzante variabili intere e variabili stringa
   AMIGADOS LATTICE C standard */

#include <stdio.h>

void main()
{
char *nome[30];
int eta, anno;

printf("Come ti chiami ? ");
scanf("%29s", nome);
printf("In che anno sei nato ? ");
scanf("%4d", &anno);
eta = 1988 - anno;
printf("Allora, %s,  hai %2d anni\n", nome, eta);
}
```

C, che, al peggio, considerano significativi i primi otto caratteri del nome, pur conservando tutti i caratteri digitati.

Esempi concreti

Per chiarirci bene le idee, vediamo qualche dichiarazione esemplificativa:

```
int luigi, luisa;
char mariella;
float paola;
float loredana;
double Luigi;
```

Supponendo l'uso del Lattice C, ciò implicherebbe che "luigi" e "luisa" siano inizialmente assunte come variabili intere (32 bit) con segno; "mariella" è una variabile di carattere (ovvero può contenere un solo carattere, un codice tra -128 e +127 standard ASCII), "paola" è una variabile in virgola mobile a precisione semplice (6 oppure 7 cifre di precisione, con eventuale notazione esponenziale, tipo 3.14E27).

Anche "loredana" è una variabile in virgola mobile, dichiarata su di un'altra linea sia perchè C lo consente sia perchè fa comodo per leggibilità. Comunque, le due variabili sono di tipo identico. "Luigi" è una variabile in doppia precisione, ma si noti bene che il primo carattere è maiuscolo. Il C, infatti, "vede" come differenti i caratteri maiuscoli e quelli minuscoli. Se, invece, nello stesso programma (più precisamente diremo "nel corpo della stessa funzione", quando sapremo che cosa vuole dire!) avessimo usato...

```
int sabrina;
...
...
double sabrina;
```

....il compilatore segnalerebbe prontamente un errore a causa della ridefinizione di una stessa variabile (operazione non ammessa); marcando questo errore come "fatale" non avrebbe generato alcun codice oggetto in uscita.

Altra nota importantissima. Visto che il C effettua distinzioni tra maiuscolo e minuscolo, non è casuale, nè trascurabile, che tutti i nomi di parole riservate al linguaggio (sono 30 in Lattice), ovvero le equivalenti delle keywords del Basic (DIM, PRINT...) debbono essere obbligatoriamente scritte in minuscolo.

MUOVERSI IN FRETTA TRA GLI SPRITES

Un videogioco, contrariamente a quanto si possa pensare, risulta molto istruttivo al fine di comprendere la gestione degli sprite in Basic 7.0

di **Giancarlo Mariani**

Per ottenere uno sprite con la massima facilità è utile "entrare" in pagina grafica, con l'istruzione Graphic 1,1, e ricorrere ai vari comandi grafici (Color, Box, Char, Circle, Draw, Paint, Scale) per visualizzare la figura desiderata che, ovviamente, deve avere le dimensioni di uno sprite (24 punti di larghezza e 21 di altezza). Si può eseguire il disegno in qualsiasi zona dello schermo.

Terminato il disegno, bisogna salvarne l'immagine in una variabile stringa tramite l'istruzione "Sshape", la cui sintassi è la seguente:

SSHAPE Var$, x1, y1, xf, yf

in cui Var$ è una variabile stringa nella quale memorizzare i dati dello sprite; x1, y1 sono le coordinate del vertice superiore sinistro del rettangolo che contiene la figura; x2, y2 sono, invece, le coordinate del vertice opposto (inferiore destro).

Il rettangolo, non ci stancheremo di ripeterlo, deve tassativamente avere dimensioni di 24x21 pixel.

Bisogna ora trovare il modo per trasferire la figura Var$ in uno sprite. A questo provvede l'istruzione "Sprsav", la cui sintassi è la seguente:

in cui var$ è la stessa variabile stringa usata poco fa nell'istruzione Sshape, mentre Sn è il numero dello sprite, nel quale trasferire i dati, che può variare tra 1 e 8 (ricordiamo che il C/128 può gestire 8 sprites diversi contemporaneamente).

Nonostante sia stato costruito il nostro bravo sprite, questo non è ancora visibile sullo schermo: dobbiamo, infatti, abilitarlo. L'istruzione che provvede al nostro scopo è "Sprite":

SPRITE sn, o, c, p, ex, ey, m

SCHEDA TECNICA

Videogame a scopo didattico

Hardware richiesto: C/128: non adattabile ad altri computer Commodore
Ideale l'uso di un monitor a colori
Consigliato ai principianti

Anche i programmi pubblicati in queste pagine sono contenuti nel disco "Directory" di questo mese.

in cui Sn è il numero dello sprite da visualizzare (1-8)
O accende (O=1) oppure spegne (O=0) lo sprite.
C è il colore (1-16) secondo la tabella:

1 = nero
2 = bianco
3 = rosso
4 = azzurro
5 = porpora
6 = verde
7 = blu
8 = giallo
9 = arancio
10 = marrone
11 = rosso chiaro
12 = grigio scuro
13 = grigio
14 = verde chiaro
15 = blu chiaro
16 = grigio chiaro

P determina la priorità: se P=1 lo sprite coprirà gli oggetti sui quali dovesse trovarsi a passare; se P=0, invece, lo sprite ne verrà coperto.
Ex indica l'espansione lungo l'asse x (Ex=1)
Ey indica la stessa cosa, ma per l'asse y.
M impone il multicolor (M=1); altrimenti (M=0) lo sprite apparirà in hi-res.

Una volta "acceso" lo sprite, bisogna posizionarlo tramite l'istruzione "Movspr":

MOVSPR sn, x, y

in cui Sn è il numero dello sprite da muovere sullo schermo.
X, Y sono le coordinate (relative allo spigolo superiore sinistro) nelle quali verrà spostato lo sprite che risulterà visibile solo per X compreso tra 24 e 344 ed Y compreso tra 50 e 250 (320x200 punti).

L'istruzione Movspr ha anche un'altra interessante sintassi:

MOVSPR sn, ang v

in cui Sn è ancora relativa al numero dello sprite da muovere.
Ang è l'angolo (compreso tra 0 e 360 gradi) con il quale si deve muovere lo sprite.
V è la velocità di movimento (tra 0 e 15).

Quest'ultima istruzione, in parole povere, fa muovere perennemente lo sprite secondo una direzione, ed una velocità, scelte senza ricorrere a cicli For - Next all'interno del programma. Una volta impartita un'istruzione Movspr, lo sprite inizia a muoversi, dall'ultima posizione che aveva assunto, secondo l'angolo Ang e la velocità V; si fermerà solo quando viene impartita un'altra istruzione simile, ma con velocità V=0.

Applicazioni pratiche

Vediamo in pratica un programmino che consenta di verificare i concetti appena visti. Accendete il C/128 (naturalmente in modo 128) e battete il listato n.1 (fino alla riga 200) che costruisce uno sprite rettangolare, lo posiziona al centro dello schermo e lo fa muovere orizzontalmente. Per fermare il movimento bisogna premere i tasti Run-Stop e Restore.

Le istruzioni viste sinora permettono di gestire, con la massima facilità, la costruzione, il posizionamento ed il movimento degli sprites, ma per creare programmi più complessi, come i videogiochi, bisogna conoscere altri comandi che consentano di rilevare le collisioni tra gli sprite, oppure tra uno sprite e lo sfondo.

Quando si intende realizzare un gioco che ricorra all'uso di sprites, è necessario sapere quando collidano tra loro oppure con lo sfondo.

Ad esempio, supponendo di avere usato uno sprite per disegnare una bomba ed un altro sprite per rappre-

Sprite e C/128

Dopo molti anni dall'uscita del C/64, quasi tutti conoscono le ottime qualità grafiche del computer, per non parlar degli sprites.

Uno sprite è una piccola figura, formata da 24x21 pixel (punti-video), che può essere spostata, colorata, espansa in qualsiasi punto dello schermo.

Il vantaggio degli sprites, rispetto alla gestione della pagina grafica, è che il processore video del computer provvede a visualizzare, ed a spostare, la figura tramite semplici comandi, senza ricorrere a complicati spostamenti di zone di pagina grafica, come è invece indispensabile fare con il C/16 o con il Plus/4.

Altro vantaggio è che le figure sono completamente indipendenti dallo sfondo; ne consegue che possono essere spostate al di sopra di messaggi o disegni, eventualmente presenti sullo schermo, senza preoccuparsi di cancellare e ridisegnare le zone "invase", dal momento che a tale operazione provvede lo stesso processore.

Sul C/64 la gestione degli sprites viene effettuata attraverso istruzioni Poke, perchè il Basic 2.0 non possiede comandi dedicati. Tale inconveniente è stato eliminato nel C/128, grazie all'interprete Basic 7.0, provvisto di tutte le istruzioni necessarie per una completa (e comoda) gestione degli sprites.
Prima di iniziare a lavorare con gli sprite, bisogna partire dalla "costruzione" dello stesso sprite, seguendo tre metodi diversi:

1 - Usare l'istruzione "Sprite"
2 - Usare l'istruzione "Sprdef"
3 - Procedere come nel C/64.

In questo articolo esamineremo solo il primo dei tre metodi; il secondo, infatti, è poco pratico da illustrare nelle pagine di una rivista, mentre il terzo, per la sua complessità, viene trascurato per ovvi motivi (alrimenti a che servirebbe avere un C/128?).

```
100 REM LISTATO DIMOSTRATIVO MOVIMENTO SPRITE
110 GRAPHIC 1.1 :REM GRAFICA HI-RES E CANCELLA LO SCHERMO
120 REM RETTANG. PIENO DI VERTICI (30.30) E (53.50) (24X21 PUNTI)
130 BOX 1,30,30,53,50,0,1
140 SSHAPE A$,30,30,53,50 :REM SALVA IL DISEGNO NELLA VARIABILE A$
150 SPRSAV A$,1 :REM TRASFERISCE I DATI DEL DISEGNO IN SPRITE 1
160 GRAPHIC 0:SCNCLR :REM MODO TESTO E CANCELLAZIONE SCHERMO
170 SPRITE 1,1,2,0,0,0,0 :REM COLORA SPRITE 1 CON COLORE BIANCO
180 MOVSPR 1,160,100 :REM POSIZIONA SPRITE AL CENTRO SCHERMO
190 MOVSPR 1,91 #7 :REM MUOVE SPRITE ORIZZONTALMENTE SULLO SCHERMO
200 END: REM FINE PRIMA PARTE
210 SPRSAV A$,2 :REM NUOVO SPRITE A FORMA DI RETTANGOLO PIENO
220 MOVSPR 2,200,150 :REM POSIZIONA IL SECONDO SPRITE
230 SPRITE 2,1,1,0,0,0,0 :REM ACCENDE LO SPRITE N.2 IN NERO
240 IF BUMP(1)=0 THEN 240 :REM CONTROLLA COLLISIONE SPRITES
250 PRINT"COLLISIONE!"
260 MOVSPR 1,91 #0 :REM FERMA IL PRIMO SPRITE
270 END
```

sentare un Ufo, è indispensabile sapere quando la bomba urta l'Ufo, per prendere varie decisioni, quali incrementare il punteggio, interrompere la partita e simili.

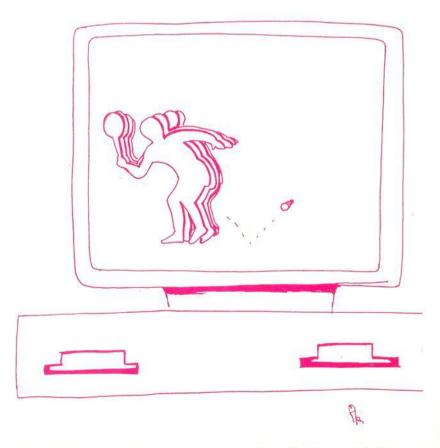
Per questo scopo vengono in aiuto due potenti istruzioni:

La prima si chiama Bump, che non è un'istruzione ma una funzione (cioè restituisce un valore, tipo SIN, ABS, ecc.)

Bump ha due modi di funzionamento:
BUMP (1) rileva le collisioni tra sprite e sprite.
BUMP (2) rileva le collisioni tra sprite e sfondo.

La funzione restituisce un valore in base al quale è possibile determinare quanti e quali sprites sono entrati in collisione con "qualcosa". Siccome sappiamo che gli sprites sono 8 (quanti sono i bit che compongono un byte) il computer associa, ad ogni sprite, il corrispondente bit (vedi tabella).

Il C/128, in altre parole, assegna sempre il valore 1 ad ogni bit corrispondente allo sprite che collide con qualcosa, mentre pone a 0 i bit relativi agli altri.

Se, ad esempio, gli sprites n.7 e n.3 collidono tra di loro, i bit 6 e 2 verranno automaticamente posti ad 1, mentre gli altri resteranno a 0.

```
240 COLLISION 1,290:REM ABIL.COLLIS.TRA SPRITE E MANDA A RIGA 290
250 IF C=0 THEN GOTO 250
260 COLLISION 1,270 :REM DISABILITA INTERRUPT DELLE COLLISIONI
270 MOVSPR 1,91 #0 :REM FERMA LO SPRITE
280 PRINT"COLLISIONE! ":END
290 IF BUMP(1)=3 THEN C=1 :REM CONTROLLA COLLIS.SPRITE 1 E 2
300 RETURN
```

Trasformando in decimale il valore binario 01000100, si ottiene il valore 68 (64+4). Questo sarà proprio il valore restituito dalla funzione BUMP (1) nel caso di avvenuta collisione tra lo sprite 3 ed il 7. Un'istruzione del tipo...

If Bump(1)=68 Then......

...consentirà il riconoscimento dell'avvenuta collisione.

Bump, con parametro 2, funziona in modo analogo, rilevando eventuali collisioni tra sprite e sfondo; ammettendo che lo sprite 4 collida con un messaggio presente sullo schermo, il C/128 provvederà immediatamente ad "alzare" ad 1 il bit 3 (binario 00001000) che, tradotto in decimale, vale 2 exp 3=8.

in un programma, quindi, l'istruzione da inserire sarà del tipo:

If Bump(2)=8 Then...

```
   Bit: 7 6 5 4 3 2 1 0
Sprite: 8 7 6 5 4 3 2 1

Valore: 0 1 0 0 0 1 0 0

2↑6*1 + 2↑2*1 = 68
```

Il valore da verificare con l'istruzione Bump si ricava sommando le potenze di due corrispondenti ai bit posti ad uno, trascurando le potenze dei bit nulli. Si noti che i bit di un byte sono, di solito numerati da 0 a 7. Gli sprite, invece, vengono numerati da 1 a 8.

Per verificare la nuova funzione, terminate di digitare il programma visto prima, che crea un altro sprite uguale al precedente e rileva la collisione tra i due. Aggiungete le righe che mancano (tranne la 200).

Si noti che il registro di collisione "ricorda" sempre l'avvenuta collisione in modo totalmente automatico e del tutto indipendente dalla volontà del programmatore.

Ciò significa che se, per esempio, avviene una collisione tra due sprite, il registro conserverà in memoria tale evento anche se, in seguito, i due sprite si sono allontanati tra loro. Se, quindi, si "interroga" il registro, questo potrà dare in risposta, con vostra sorpresa, una segnalazione di collisione, anche se, al momento "attuale", questa non è verificata.

In altre parole: la stessa operazione di lettura del registro (If Bump... eccetera) provvede a fornire l'informazione richiesta e, nello stesso momento, ad azzerare il registro, rendendolo immediatamente disponibile per memorizzare altre collisioni, che conserverà gelosamente fino alla successiva lettura.

Un altro sistema

L'altra potente istruzione che permette di rilevare collisioni è "Collision".

Collision1,LN
permette di rilevare collisioni tra sprite e sprite

Collision2,LN
controlla le collisioni tra sprite e sfondo.

La sostanziale differenza tra questa istruzione e quella vista precedentemente, è che genera un controllo direttamente in Interrupt.

Se viene verificata una collisione, il

UN'EMOZIONE DA 1200 BIT AL SECONDO

- La potenza di una banca dati, la dinamica di un quotidiano.
- L'unico servizio telematico italiano con le notizie in tempo reale sul mondo dell'informatica.
- Il solo accessibile tramite la rete nazionale Videotel presente in piu' di 32 distretti telefonici (oltre 1000 comuni!).
- Con LASERNET 800 potrai caricare programmi in TELESOFTWARE, chiacchierare in diretta con tutta Italia sulle CHATLINES, editare un tuo spazio personale su PRIMA PAGINA, leggere le notizie piu' interessanti di LASER NEWS e migliorare la tua programmazione con i nostri corsi.
- Oltre 5000 pagine consultabili 24 ore su 24.
- Il nostro servizio ti costa ogni giorno meno della meta' di un quotidiano!

Per avere maggiori informazioni sul servizio compila il tagliando e spediscilo a:
LASERNET 800
VIA G. MODENA, 9
20129 MILANO - T.02/208201

PROVALA!

Desidero ricevere maggiori informazioni su LASERNET 800

Cognome............... Nome...........
Via..................................
Citta'.........................Prov....
CAP........... TEL...../.............
Data di nascita .../.../...

Il mio computer e' un:
Commodore ☐64 ☐128 ☐Amiga
☐MSX ☐BBC ☐Atari ST ☐PC
Spectrum ☐48K ☐Plus ☐128
☐ Ho gia' un adattatore telematico

CCC

programma salta alla linea specificata da Ln.

L'istruzione Collision va impartita una volta sola all'inizio del programma; in seguito, durante l'elaborazione, ogni volta che uno sprite collide-rà con sfondo o con altro sprite, il programma salterà automaticamente alla linea Ln.

Poichè il salto equivale ad un normalissimo GOSUB, alla fine della routine, che parte da Ln, dovrà esserci il comando Return.

Collision permette di saltare ad una routine specifica ma, purtroppo, non consente di determinare quanti e quali sprites sono entrati in contatto; alla carenza indicata si provvede inserendo, nella stessa subroutine, istruzioni tipo Bump.

Si tenga presente che Collision, attivata priva del numero di linea, disabilita l'interrupt; tuttavia, per motivi che non stiamo qui ad illustrare, è preferibile indicare sempre una linea Bas c terminale, cui indirizzare l'interrupt.

Lasciando inalterate le linee da 0 a 120, aggiungete ora al programmino di esempio le righe pubblicate a parte (240-300).

E' da notare che, specialmente a velocità elevate, a causa della relativa lentezza del Basic, lo sprite riesce a superare di poco l'ostacolo prima che la collisione venga rilevata.

Per finire..

Il semplice gioco, proposto a scopo puramente didattico, viene consigliato a tutti coloro desiderino approfondire l'argomento.

Il listato è commentato in modo da far comprendere al lettore le tecniche usate, che peraltro sono le stesse spiegate nell'articolo.

```
100 REM ** UFO GAME
110 REM ** (DIMOSTRATIVO DEI COMANDI
120 REM ** DEGLI SPRITES DEL 128)
130 REM ** BY MARIANI G.
140 :
150 SCNCLR:GOSUB 650:REM ** DEF.SPR.
160 SCNCLR:
170 PRINT "■■■ UFO-GAME BY MARIANI G."
180 PRINT:PRINT" A=SIN., D=DES., SPAZIO=FIRE"
190 PRINT:PRINT" DIFFICOLTA' (1-15)"
200 INPUT D:IF D<0 OR D>15 THEN 160
210 V=D:IF D<5 THEN V=5:REM **VEL. PROIETTILE
220 P=0:SCNCLR:FOR K=1 TO 40:PRINT"■ ■";:NEXT:PRINT"■"P:E=0
230 X1=160:Y1=230:REM ** COORD. CANNONE
240 X2=30:Y2=60:REM ** COORD. UFO
250 MOVSPR 1,X1,Y1:REM ** POSIZIONA CANNONE
260 MOVSPR 3,X2,Y2:REM ** POSIZIONA UFO
270 SPRITE 1,1,2,0,0,0,0:REM ** ABILITA CANNONE
280 SPRITE 3,1,2,0,0,0,0:REM ** ABILITA UFO
290 COLLISION 1,580:REM ** ABILITA COLL. SPRITE-SPRITE
300 MOVSPR 3,91 #D
310 GET A$
320 IF A$="A" AND X1>30 THEN X1=X1-3:MOVSPR 1,X1,Y1
330 IF A$="D" AND X1<320 THEN X1=X1+3:MOVSPR 1,X1,Y1
340 IF A$=" " THEN GOSUB 530
350 A$=""
360 REM ** DET. COLLISIONE DI PROIETTILE CON LO SFONDO
370 REM DISABILITA PROIETTILE
380 IF BUMP (2)=2 THEN MOVSPR 2,360 #0:SPRITE 2,0,2,0,0,0,0
390 IF BUMP(1)=5 OR E<>0 OR P>=10 THEN 410:REM ** FINE GIOCO
400 GOTO 310
410 REM ** FINE GIOCO
420 MOVSPR 2,360 #0:MOVSPR 3,91 #0:REM ** FERMA PROIETTILE E UFO
430 FOR K=1 TO 1500:NEXT
```

```
440 REM ** DISABILITA SPRITES
450 SPRITE 1,0,2,0,0,0,0:SPRITE 2,0,2,0,0,0,0:SPRITE 3,0,2,0,0,0,0
460 SCNCLR
470 PRINT:PRINT" PUNTI:"P
480 IF P>=10 THEN PRINT"■■ BRAVO, HAI VINTO!!"
490 PRINT:PRINT" PREMI RETURN PER INIZIARE"
500 INPUT A$
510 GOTO 160
520 END
530 REM ** SPARA PROIETTILE
540 MOVSPR 2,X1,Y1-17:REM ** POSIZIONA PROIETTILE
550 SPRITE 2,1,2,0,0,0,0:REM ** ABILITA PROIETTILE
560 MOVSPR 2,360 #V:REM ** FA MUOVERE PROIETTILE
570 RETURN
580 REM ** COLLISIONE SPRITE-SPRITE
590 REM PROIETTILE-UFO: INCR. PUNTI E DISABILITA PROIETTILE
600 IF BUMP (1)=6 THEN 630
610 IF BUMP (1)=5 THEN E=1:REM ** COLLISIONE UFO-CANNONE
620 RETURN
630 P=P+1:PRINT"S"P:MOVSPR 2,360 #0:SPRITE 2,0,2,0,0,0,0
640 GOTO 610
650 REM ** DEFINIZIONE SPRITES **
660 REM ← CANNONE
670 GRAPHIC 1,1
680 DRAW 1,33,27 TO 30,30
690 DRAW TO 30,34
700 DRAW TO 45,34
710 DRAW TO 45,30
720 DRAW TO 42,27
730 DRAW TO 33,27
740 BOX 1,37,27,38,22
750 SSHAPE A$,26,18,49,38
760 SPRSAV A$,1
770 REM ← PROIETTILE
780 DRAW 1,27,60 TO 30,57
790 DRAW TO 33,60
800 DRAW 1,30,57 TO 30,53
810 SSHAPE B$,19,60,42,40
820 SPRSAV B$,2
830 REM ← UFO
840 DRAW 1,30,100 TO 44,100
850 DRAW TO 40,95
860 DRAW TO 34,95
870 DRAW TO 30,100
880 CIRCLE 1,37,95,3,3,0,90
890 CIRCLE 1,37,95,3,3,270,360
900 SSHAPE C$,26,106,49,86
910 SPRSAV C$,3
920 A$="":B$="":C$=""
930 GRAPHIC 0:RETURN
940 END
```

CAMPUS

LABORATORIO SOFTWARE DI COMMODORE COMPUTER CLUB

UN GRANDE INSERTO DA COLLEZIONARE

PRINCIPIANTI

- Basta la parola
- Il programma è servito
- La terza via per leggere la directory di un disco

CAMPIONI

- Tutti Spielberg con il Commodore 64

ESPERTI

- L'Assembly, questo sconosciuto
- Reverse window per C/128
- Un comando in più per il Monitor del 128
- Animazioni ultrarapide per 128

La Grande Libreria Systems

Autori Vari

64 Programmi per Commodore 64

Giochi, grafica, gestione delle stringhe, musica, numeri, gestionali.

Lire 4.800

Autori Vari

I miei amici C16 & Plus4

Un manuale pratico per padroneggiare il basic di questi computer.

Lire 7.000

Autori Vari

Strategie vincenti per Commodore 64

Le strategie per tutti i classici del videogioco: per giocarli, vincerli o programmarli.

Lire 5.800

Autori Vari

62 Programmi per il Vic 20, C16 e Plus 4

Giochi, grafica e routine per imparare a programmare.

6.500

Roberto Didoni, Guido Grassi

Utilities e giochi didattici

Raccolta di programmi pratici per tutti i Commodore e lo Spectrum.

Lire 6.500

Giovanni Mellina

Tutti i segreti dello Spectrum

4 passi nella Rom: come usare le più importanti routine del sistema operativo.

Lire 7.000

Roberto Didoni, Guido Grassi

Simulazioni e test per la didattica

Teoria e listati per Vic 20, C16, C64 C128 e Spectrum Sinclair.

Lire 7.000

Paolo Goglio

Impara giocando il basic dello Spectrum

Esercizi pratici per entrare nel vivo della programmazione.

Lire 7.000

Clizio Merli

µPascal per Commodore 64/128

Un manuale completo per il programma compilatore

Lire 7.000

Umberto Colapicchioni e Luca Galuzzi

Dal registratore al drive del C64

Tutti i segreti delle memorie di massa del Commodore 64

Lire 7.000

Autori Vari

ADA

Il linguaggio passepartout dei computer degli anni '80.

Lire 5.000

Clizio Merli

Il linguaggio PASCAL

Un manuale tascabile per lo studio e la programmazione.

Lire 5.000

Sì, voglio arricchire la mia biblioteca con i seguenti volumi al prezzo di copertina + lire 3.000 per spese di spedizione.

- ☐ **64 Programmi per Commodore 64**
- ☐ **Strategie vincenti per i tuoi videogames**
- ☐ **62 Programmi per Vic 20 C16 e Plus77**
- ☐ **Utilities e giochi didattici**
- ☐ **Tutti i segreti dello Spectrum**
- ☐ **Sìmulazioni e test per la didattica**
- ☐ **Imparare giocando il basic dello Spectrum**
- ☐ **I miei amici C16 e Plus4**
- ☐ **Pascal per Commodore 128**
- ☐ **Dal registrattore al drive del C64**
- ☐ **ADA**
- ☐ **Il linguaggio Pascal**

Nome ....

via .... **N.ro.** .... **telefono** ....

CAP .... **Città** ....

Su tale importo mi praticherete lo sconto del 10% in quanto abbonato a ☐ *Commodore Computer Club* ☐ *Personal Computer* ☐ *Com puter* ☐ *VR Videoregistrare*. Pertanto vi invio la somma soltanto di lire ....

Valore dell'ordine lire..

Ritagliare e spedire in busta chiusa regolarmente affrancata a Systems Editoriale - V.le Famagosta, 75 - 20142 Milano.

BASTA LA PAROLA!

Chi legge per la prima volta libri e riviste di informatica spesso trova difficoltà a comprendere alcuni termini

di **Alessandro de Simone**

Numerosi lettori, soprattutto alle prime armi, chiedono spesso di spiegare la terminologia informatica.

Naturalmente non ci è possibile pubblicare un dizionario specifico, sia perchè lo spazio a disposizione non lo consente, sia perchè vi sono già molti volumi in commercio.

Chi volesse procurarsi, presso in nostro servizio arretrati (tel. 02/84.67.34.8) il "Vocabolario inglese / italiano dell'informatica", troverà la terminologia ufficiale ANSI di numerosi termini specifici. Gli altri, invece, possono certamente accontentarsi di esaminare, di tanto in tanto, le note esplicative relative alle parole più usate, che abbiamo intenzione di riportare saltuariamente su queste stesse pagine.

Allocare

In qualsiasi linguaggio vi sono istruzioni e dati da "trattare" per portare a termine un'elaborazione.

Tali dati, o una gran parte di essi, devono necessariamente esser presenti nella memoria del calcolatore incaricato di svolgere il compito.

Pertanto lo strumento software (brutalmente: il programma) che legge da tastiera, da disco o da un qualsiasi altro strumento similare (modem, penna luminosa, tavoletta grafica e così via) deve trasferire, all'interno del calcolatore, i dati necessari per compiere, in seguito, l'elaborazione.

Se, ad esempio, indichiamo con A$ la stringa destinata ad ospitare la parola "computer", è necessario trasferire, cioè "allocare", i singoli caratteri (c-o-m-p-u-t-e-r) in altrettanti byte della memoria Ram.

Lo stesso termine, "allocare", si utilizza soprattutto per indicare l'operazione analoga di scrittura su disco; in questo caso la differenza consiste nel fatto che, anzichè su memoria Ram, i dati vengono allocati su minuscole zone del supporto magnetico.

Il termine de-allocare può sembrare che si riferisca all'operazione contraria. Indica, invece, l'operazione che rende disponibili, al sistema operativo del disco (Dos), alcune zone magnetiche precedentemente occupate da dati o programmi.

Quando, in altre parole, si chiede al Dos di cancellare un programma presente sul disco, l'operazione non viene compiuta cancellando fisicamente dato per dato: il Dos, infatti, semplifica le operazioni "deallocando" le zone stesse, informando, cioè, la directory (indice del disco) che la zona magnetica indicata, nonostante contenga ancora dei dati, deve ritenersi libera per ospitare altri dati o programmi.

Scrivere, allocare, memorizzare, trascrivere sono termini che spesso indicano la stessa cosa

Le "famiglie" in lotta tra loro sono, attualmente, quella del 68000 (Amiga) e dell'80386 (IBM)

Editing

E' l'operazione generica con cui si indica l'azione di scrivere correttamente un testo, oppure un programma, secondo le regole richieste dal software usato.

Si effettua "Editig" quando, scrivendo un programma in Basic, si digita il numero di linea, si ricorre alle abbreviazioni di alcune istruzioni, si chiede di listare il programma (in tutto o in parte), si renumera (se possibile), si duplicano o si cancellano linee.

Si "edita" un testo con un word processor quando, ricorrendo alle potenzialità del software, si inseriscono linee, si modificano parole battute erroneamente, si trasformano caratteri minuscoli in maiuscoli (o viceversa), si cancellano o si spostano frasi all'interno di un testo e così via.

Famiglia

Nulla a che fare con la mafia ha questo termine, tipicamente americano, con cui si desidera indicare un gruppo di circuiti integrati (i famosi "chip") ognuno dotato di caratteristiche particolari, ma fabbricati in modo da ottimizzare le risorse di un sistema computerizzato cui appartengono.

I "rami" di una famiglia si possono distinguere in orizzontali e verticali.

Consideriamo, ad esempio, i microprocessori della famiglia 6502. Il primo micro, che aveva questa sigla, fu montato sui gloriosi Commodore PET, sul Vic-20 e sui primi modelli della serie Apple.

In seguito, grazie ai progressi della microelettronica (ed alle esigenze dei progettisti di computer) furono realizzati micro più potenti ma, tutto sommato, basati sulla stessa struttura dei micro precedenti.

Nacque, così, il 6510 che fu adottato nel C/64. Si differenziava dal precedente 6502 per la possibilità di gestire due banchi di memoria, operazione indispensabile per selezionare, nel C/64, la memoria Ram oppure Rom.

In seguito nacque l'8502, adottato per il C/128 che, oltre a funzionare in modo perfettamente identico al 6510 (e, a maggior ragione, al 6502) consentiva la gestione di un maggior numero di banchi di memoria.

Mentre, quindi, la "famiglia" si estende verticalmente con il progredire della potenza del micro, si può estendere anche "orizzontalmente", nel senso che vengono prodotti chip specializzati in I/O (ingresso e uscita dei dati) come il 6526; oppure in gestione del suono, come il famoso SID, sintetizzatore del C/64 (6581), o nello scambio di informazioni con il video (VIC 6567).

La famiglia 65XX, pertanto, indica il gruppo di circuiti integrati assortiti in modo da sfruttare al massimo, reciprocamente, le risorse di cui dispongono.

Attuale

Spesso viene usato questo termine per indicare la situazione che può esser presente in un momento molto particolare dell'elaborazione in corso.

Consideriamo, ad esempio, la seguente espressione:

"La posizione attuale dello sprite n.1 viene memorizzata nelle locazioni 53250 e 53251"

In parole più semplici significa che, durante l'elaborazione di una routine, i numeri di riga e di colonna in cui è visualizzato lo sprite vengono costantemente aggiornati ed il loro valore riportato nelle locazioni citate.

In qualsiasi momento, pertanto, la lettura del contenuto di tali locazioni fornirà le informazioni sulla posizione dello sprite nel momento esatto in cui l'informazione stessa viene richiesta. In momenti successivi, infatti, il contenuto può esser diverso.

Interfaccia

Spesso capita di dover trasferire alcuni dati da un calcolatore ad un altro oppure ad un apparecchio che non sia un computer (drive, registratore, stampante, modem e così via).

Dal momento che, per esigenze costruttive, non è possibile ricorrere ad uno stesso standard, è necessario utilizzare particolari apparecchi (le interfacce, appunto) che si incaricano di adattare le esigenze dell'apparecchio trasmittente a quello del circuito ricevente.

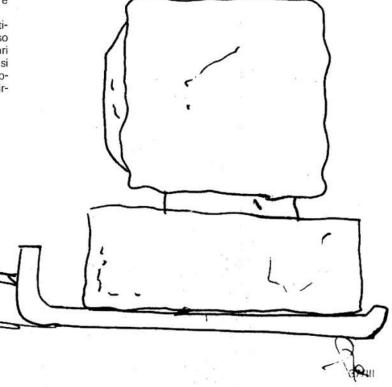
Tutti i computer dispongono di almeno un'interfaccia; ricordiamo il C/64 che possiede quella seriale (cui si connettono il drive e la stampante) e la RS-232 per altre applicazioni.

Con il termine interfaccia, inoltre, si usa spesso indicare un programma che consente di mettere in collegamento due package diversi.

Tra l'altro, il linguaggio interprete Basic è spesso considerato un'interfaccia utente, nel senso che, consentendo l'utilizzo di semplici parole inglesi, permette ad un utente, anche inesperto, di programmare una struttura elettronica, relativamente complessa, come un computer.

Monitor

Lavorando in linguaggio macchina (l.m.) capita di dover verificare il contenuto di alcune locazioni di memoria che rappresentano il codice operativo.

Per facilitare il compito dell'operatore, sono stati realizzati numerosi programmi dal nome generico di "Monitor". Questo termine, infatti, ha a che fare con il notissimo tubo a raggi catodici solo per l'utilizzo che si

fa di un apparecchio elettronico del genere.

"Monitor", quindi, è non un programma che provvede a trattare routine grafiche (come si potrebbe erroneamente supporre) ma un'utility incaricata di eseguire il monitoraggio della memoria del computer su cui gira, cioè a svolgere l'esame approfondito di aree di memoria (e del loro significato) altrimenti impossibili da esaminare con altri strumenti.

Versione

Il latino o il greco non c'entrano con le versioni di cui stiamo per parlare.

Con tale termine, infatti, ci si riferisce all'edizione di un programma oppure al modello di un computer presente sul mercato.

Spesso, dopo aver commercializzato programmi o sistemi operativi, ci si accorge di alcuni bug (:errori) di funzionamento oppure alla mancanza di un'opzione che potrebbe rivelarsi utile.

La software house, pertanto, modifica il programma che, dopo un certo tempo, ripropone potenziato e migliorato.

Di solito le versioni successive di un package vengono vendute a prezzi più bassi se si dimostra di aver acquistato il package precedente; quasi sempre, poi, i file elaborati dalle versioni precedenti possono esser "trattati" da quelle successive (ma non viceversa).

A volte, infine, si trovano più versioni di uno stesso programma che si differenziano tra loro a seconda del computer cui sono destinate, della configurazione minima di memoria, della disponibilità, o meno, di un monitor a colori, di un doppio drive e così via.

IL PROGRAMMA E' SERVITO

Semplici considerazioni sulla gestione di un programma Basic mediante l'uso dei menu

di **Alessandro de Simone**

Non sempre un programma si limita ad elaborare soltanto una serie di calcoli fornendo, alla fine, un unico risultato.

Molto spesso, infatti, capita di dover seguire un percorso logico piuttosto che un altro e quasi mai, in questi casi, la decisione da prendere può essere automatizzata.

Se, ad esempio, volete divertirvi a scrivere un programma di geometria, potete seguire due strade: scrivere tanti programmi quanti sono i problemi da affrontare (area di un triangolo, perimetro di un rettangolo, superficie laterale di un cilindro, suo volume, eccetera) oppure scriverne uno solo che, appena lanciato con il solito Run, vi chieda quale problema risolvere.

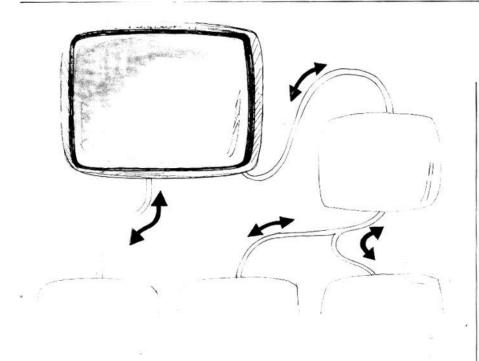
Nel primo caso si può ottenere qualche facilitazione di scrittura, oltre al vantaggio di non esser costretti ad effettuare "salti" da una parte all'altra del listato; c'è però lo svantaggio di esser costretti a caricare il programma specifico (cancellando inevitabilmente quello presente in memoria) tutte le volte che volete risolvere un problema diverso.

Il secondo metodo, quello a menu, offre innegabili vantaggi di uso ma sembra presentare, di contro, alcuni problemi di strutturazione.

Fortunatamente ci vengono incontro alcune istruzioni tipiche del Basic (If...Then, Gosub...Return) che, opportunamente utilizzate, permettono una notevole versatilità di programmazione.

In questo articolo viene illustrato un esempio applicativo che mostrerà in che modo due programmi del tutto diversi l'uno dall'altro possano esser "fusi" tra loro e scelti servendosi di un comodo menu.

Alla fine verrà illustrato un programma più ampio che, per esser gestito correttamente, presenta numerosi menu "nidificati", ossia che si richiamano l'un l'altro e che presentano specifiche applicazioni.

Il menu presente in un programma è del tutto simile a quello che ci presentano in qualsiasi ristorante: non è altro che una scelta delle varie pietanze (elaborazioni) disponibili, tra cui scegliere liberamente

Il ricorso ad un menu è quasi indispensabile anche in programmi di modeste dimensioni

Catena di moltiplicazioni

Il semplicissimo listato di questa pagina ("Moltiplicatoria") è il primo dei due listati che "fonderemo" tra loro e che sceglieremo tramite menu.

Esso consente di effettuare la moltiplicazione successiva di numeri interi. Ad esempio, la moltiplicatoria di 7, partendo dal numero 3, si sviluppa nel modo seguente:

*3*4*5*6*7 = 2520*

Se si parte dal numero 1 l'operazione prende, più propriamente, il nome di "Fattoriale". Si può notare facilmente che il valore elaborato cresce enormemente, tanto è vero che il computer non è in grado di calcolare il fattoriale di un numero maggiore di 33 (cioè 1*2*3....*33) dal momento che il valore massimo elaborabile è:

Val.Max= 1.701411833 E+38

Per evitare un segnale di "Overflow error", che si verificherebbe nel caso si tentasse di eseguire moltiplicazioni eccessive, viene utilizzato un piccolo trucco: prima di effettuare il prodotto con il fattore successivo si effettua la divisione tra il valore massimo (Z) e l'ultimo valore ottenuto; se il risultato è minore del successivo fattore (riga 150) l'operazione viene eseguita; in caso contrario viene emesso un opportuno messaggio e l'elaborazione si interrompe.

Non sempre, purtroppo, il "filtro" introdotto impedisce l'overflow, a causa di arrotondamenti, attuati con i numeri espressi in forma esponenziale, introdotti dallo stesso elaboratore.

All'inizio è presente un controllo che impedisce al programma di accettare valori negativi oppure numeri "di partenza" maggiori o eguali al numero che rappresenta il traguardo da raggiungere.

Il listato, numerato da 1,00 a 210, presenta una numerazione di linea compatibile con il successivo.

```
100 INPUT "MOLTIPLICATORIA DI";X
110 Y=1: INPUT "CALCOLO DA";Y: W=Y
120 Z=1.701411833E+38:REM VALORE MAX.
130 IF Y>=X OR Y<=0 THEN 100
140 FOR I=Y+1 TO X:
150 IF (Z/Y)<I THEN 170
160 Y=Y*I: NEXT: GOTO 200
170 PRINT"IL NUMERO"X" NON E' VALIDO".
180 PRINT"POSSO CALCOLARE DA"W
190 PRINT"FINO A"I-1
200 PRINT"VALORE:"Y: PRINT
210 GOTO 100
```

Sommatoria

Questo listato è del tutto analogo al precedente dal momento che effettua la somma di valori interi e positivi.

Il valore elaborato, stavolta, non viene incrementato con la velocità di quello precedente; se esagerate introducendo, in fase di Input, intervalli elevati tra il valore iniziale e quello finale, si possono verificare casi in cui il completamento dell'operazione richiede parecchi minuti. La sommatoria dei primi mille numeri, tanto per dare un'idea, richiede 10 secondi.

Per ciò che riguarda il resto, il programma non presenta particolari problemi; si noti, soprattutto, la numerazione adoperata: il fatto che non inizi dalla riga 220 (come forse vi aspettavate) è del tutto irrilevante.

Ciò che importa, infatti, è che inizi con un numero di linea maggiore del programma con il quale vogliamo fondere il listato. Lo spazio lasciato vuoto (220-290), anzi, potrà esser prezioso nel caso si desideri introdurre linee supplementari nel caso avessimo trascurato di considerare elaborazioni, invece, necessarie.

Chi dispone di una versione Basic che consente la renumerazione, ovviamente, non deve preoccuparsi minimamente di lasciare "spazi" vuoti di numerazione; i possessori di C/64, a differenza del C/16 e C/128, non dispongono, purtroppo, di tale utility e sono costretti, loro malgrado, a prestare la massima attenzione alla numerazione delle linee Basic.

A questo punto vi consigliamo di registare anche questo listato, in modo da possedere una versione di "sicurezza" di entrambi i programmi presentati finora.

Vedremo subito come sarà possibile effettuare una "fusione" tra due programmi e come introdurrre le semplici modifiche che consentono il salto da una parte all'altra del programma stesso in modo da svolgere l'una o l'altra elaborazione.

```
300 INPUT "SOMMATORIA DI";X
310 Y=1: INPUT "CALCOLO DA";Y: W=Y
320 Z=1.701411833E+38:REM VALORE MAX.
330 IF Y>=X OR Y<=0 THEN 300
340 FOR I=Y+1 TO X:
350 IF (Z/Y)<I THEN 370
360 Y=Y+I: NEXT: GOTO 400
370 PRINT"IL NUMERO"X" NON E' VALIDO"
380 PRINT"POSSO CALCOLARE DA"W
390 PRINT"FINO A"I
400 PRINT"VALORE:"Y: PRINT
410 GOTO 300
```

La fusione dei programmi

Siamo ora pronti per effettuare la fusione dei programmi e per aggiungere la manciata di righe che ne permetteranno un'agevole manipolazione.

Se avete ancora in memoria il secondo programma (avete, comunque, provveduto a registrarlo?), potete notare che, grazie alla sua brevità, esso viene contenuto nella parte superiore dello schermo (soltanto 16 righe, compreso il comando "List" ed il messaggio "Ready"). Al rigo successivo digitate...

Load "Moltiplicatoria"

...se, ovviamente, avete assegnato tale nome al primo listato; non dimenticate di aggiungere il suffisso ",8" se state operando con il disco.

Così facendo il programma "Sommatoria", presente in memoria prima del comando Load, viene cancellato e sostituito, appunto, con "Moltiplicatoria".

Al termine del caricamento, tuttavia, sullo schermo è ancora visualizzato il "vecchio" programma, nonostante, come abbiamo detto, non sia presente in memoria.

Stando, ora, bene attenti a non cancellare lo schermo (operazione, questa, che vi costringerebbe a ripetere tutto daccapo) posizionatevi con il cursore sulla riga 100 e premete il tasto Return di seguito, fino ad arrivare in riga 210.

Agendo in questo modo non avete fatto altro che aggiungere le 12 righe del listato "Moltiplicatoria" al programma "Sommatoria" che avete appena caricato dal supporto magnetico.

Il computer, insomma, riterrà che abbiate digitato quella dozzina di righe e, in buon ordine, le avrà aggiunte al programma in memoria.

Il trucco appena descritto, tuttavia, funziona solo se le righe da aggiungere sono in numero tale da essere visualizzate permanentemente sul video: non devono essere cancellate, quindi, da fenomeni di scrolling dovuti ai messaggi di caricamento, nè da involontarie pressioni dei tasti cursore (o, peggio, di cancellazione schermo).

Nel caso in cui i programmi da fondere siano entrambi più lunghi di una mezza videata, risulta necessario ricorrere ad opportuni funzioni di "Append", vale a dire programmi (più volte pubblicati su questa stessa rivista) che provvedono, grazie alla manipolazione di opportune informazioni, a legare fra loro un numero indefinito di listati Basic.

Dopo aver fuso i due programmi in uno solo, quindi, provvedete a digitare le righe che mancano per renderlo perfettamente identico a quello pubblicato: ci riferiamo alle righe da 10 a 90; alla 105, alla 210, alla 305, alla 410 e 420.

Le righe aggiunte servono per rendere i due listati dipendenti dal menu di scelta le

```
10 MS$="NON DISPONIBILE"
15 PRINT CHR$(147)"SCEGLI:":PRINT
20 PRINT"1- MOLTIPLICATORIA"
30 PRINT"2- SOMMATORIA"
35 PRINT"3- SOTTRATTORIA"
40 GET A$: IF A$="" THEN 40
50 IF A$="1"THEN GOSUB100:  GOTO15
60 IF A$="2"THEN GOSUB300:  GOTO15
70 IF A$="3"THEN PRINT MS$: GOTO20
80 GOTO 15
90 :
100 INPUT "MOLTIPLICATORIA DI";X
105 IF X=0 THEN RETURN
110 Y=1: INPUT "CALCOLO DA";Y: W=Y
120 Z=1.701411833E+38
130 IF Y>=X OR Y<=0 THEN 100
140 FOR I=Y+1 TO X:
150 IF (Z/Y)<I THEN 170
160 Y=Y*I: NEXT: GOTO 200
170 PRINT"IL NUMERO"X" NON E' VALIDO"
180 PRINT"POSSO CALCOLARE DA"W
190 PRINT"FINO A"I-1
200 PRINT"VALORE:"Y: PRINT
210 GOTO 100
250 :
300 INPUT "SOMMATORIA DI";X
305 IF X=0 THEN RETURN
310 Y=1: INPUT "CALCOLO DA";Y: W=Y
320 Z=1.701411833E+38
330 IF Y>=X OR Y<=0 THEN 300
340 FOR I=Y+1 TO X:
350 IF (Z/Y)<I THEN 370
360 Y=Y+I: NEXT: GOTO 400
370 PRINT"IL NUMERO"X" NON E' VALIDO"
380 PRINT"POSSO CALCOLARE DA"W
390 PRINT"FINO A"I
400 PRINT"VALORE:"Y: PRINT
410 GOTO 300
420 END
```

cui informazioni si possono rintracciare nelle righe da 10 a 80.

Se il tasto premuto è "1" (vedi riga 50) verrà attivata la prima subroutine (100-210); la seconda, invece, sarà richiamata dalla pressione del tasto "2" (riga 60).

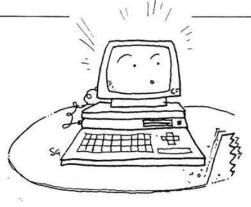
Si "ritorna" da una delle due subroutine rispondendo con un valore nullo in fase di Input (righe 105, 305). Anche questo, se vogliamo, può essere considerato un espediente per effettuare "salti" all'indietro: invece di limitarsi a non accettare digitazioni errate, si approfitta dell'occasione per considerare l'errata digitazione come un "segnale" per tornare al menu principale.

Si noti che, subito dopo il comando di Gosub (righe 50, 60) è presente un perentorio "Goto 15" che obbliga il computer a riprendere l'elaborazione dall'inizio.

Importante, infatti, è stabilire un percorso inequivocabile per l'elaborazione e mettersi nei panni del computer che, dopo aver soddisfatto il Gosub (e tornato indietro grazie al comando Return) elabora l'istruzione immediatamente successiva allo stesso Gosub. Nulla di meglio, quindi, che imporre un funzionale Goto indirizzato verso l'inizio del menu che, quasi sempre, coincide con l'inizio del programma.

Il quarto programma

L'ultimo programma di queste pagine,che invitiamo i lettori volenterosi a digitare e ad osservare con la massima attenzione, consente di

- *Introdurre un certo numero di valori.*
- *Determinarne la media aritmetica.*
- *Inviare i dati digitati su drive, stampante oppure registratore.*
- *Richiamare, da nastro o disco, i dati precedentemente memorizzati.*

Si può notare che alcuni menu richiamano altri menu e, alcuni di questi, ne richiamano altri ancora. Tutti, però, dispongono della stessa struttura: un comando "Gosub", il corrispondente "Return" ed un "Goto" posizionato immediatamente dopo il "Gosub" (righe 230, 580); altre subroutine sono gestite in modo analogo, pur se non del tutto identico a quello descritto.

Interesse può destare la tecnica usata per fare in modo di rendere operative, in particolari momenti dell'elaborazione, solo determinate scelte, e non altre.

Il menu può richiamare un altro menu che, a sua volta, può proseguire in "cascata"

All'inizio, infatti, è possibile soltanto inizializzare il vettore numerico atto a contenere i valori di cui si vuol determinare la media, oppure caricare, da supporto magnetico, un gruppo di dati precedentemente registrati. Non ha senso, infatti, chiedere di inserire dati in un vettore ancora da definire (Dim) oppure riversare, su stampante, dati inesistenti.

Analogamente, dopo aver digitato alcuni dati, deve essere impedita l'operazione di caricamento di un file che distruggerebbe i dati faticosamente digitati.

A tutto ciò provvedono le variabili-interruttori X(1), X(2)... X(5) che opportunamente definite con valori nulli o unitari, impediscono alcune elaborazioni (righe 230-270) oppure ne consentono altre (righe 590-610; 370-380).

N.B: non abbreviare il comando Print con il punto di domanda (?), pena segnalazione di Syntax error.

```
100 REM PROGRAMMA IDONEO PER: C/16, C/64, C/128, PLUS/4, VIC/20
110 REM PER COMPRENDERE LA STRUTTURA A MENU DI UN PROGRAMMA
120 REM CONTIENE MENU E SUB-MENU
130 :
140 X$=CHR$(18)+" (IMPOSSIBILE)":DI=1
150 PRINT CHR$(147)"SCEGLI:"
160 PRINT:PRINT"1- INIZIALIZZARE   ";:IF X(1)=1 THEN PRINT X$;
170 PRINT:PRINT"2- INSERIRE VALORI";:IF X(2)=0 THEN PRINT X$;
180 PRINT:PRINT"3- MEDIA ARITMET. ";:IF X(3)=0 THEN PRINT X$;
190 PRINT:PRINT"4- CARICARE DATI   ";:IF X(4)=1 THEN PRINT X$;
200 PRINT:PRINT"5- INVIARE DATI    ";:IF X(5)=0 THEN PRINT X$;
210 IF DI>1 THENPRINT:PRINT:PRINTCHR$(18)"DATI INSERITI:"DI-1
```

```
220 GOSUB 800
230 IF A$="1" AND X(1)=0 THEN GOSUB 300: GOTO 150
240 IF A$="2" AND X(2)=1 THEN GOSUB 410: GOTO 150
250 IF A$="3" AND X(3)=1 THEN GOSUB 480: GOTO 150
260 IF A$="4" AND X(4)=0 THEN GOSUB 320: GOTO 150
270 IF A$="5" AND X(5)=1 THEN GOSUB 520: GOTO 150
280 GOTO 150:REM "RINVIO" DI SICUREZZA
290 :
300 PRINT CHR$(147): INPUT"QUANTI DATI RITIENI DI INSERIRE";ND
310 DIM Y(ND):X(1)=1:X(2)=1:X(4)=1:RETURN
320 PRINT CHR$(147)"SCEGLI:":PRINT"1-REGISTRATORE":PRINT"2-DRIVE"
330 GOSUB 800:IF A<1 OR A>2 THEN RETURN
340 DN=1:IF A=2 THEN DN=8
350 INPUT"NOME DEL FILE";NO$:IF LEN(NO$)=0 AND A=2 THEN RETURN
360 X(1)=1:X(2)=1:X(3)=1:X(4)=1:X(5)=1
370 IF DN=8 THEN OPEN 8,8,8,NO$+",S,R"
380 IF DN=1THEN OPEN 8,1,0,NO$
390 INPUT#8,ND: INPUT#8,DI:DIM Y(ND)
400 FOR I=1 TO DI-1:INPUT#8,Y(I):NEXT:CLOSE 8:RETURN
410 PRINT CHR$(147)CHR$(18)"(DIGITA: * PER RITORNARE AL MENU)"
420 PRINT
430 IF DI>ND THEN X(2)=0:PRINT"VETTORE ESAURITO":GOSUB 800:RETURN
440 PRINT DI;:INPUT"DATO";Y$:IF Y$="*" THEN RETURN
450 Y(DI)=VAL(Y$): DI=DI+1
460 IF DI>0 THEN X(3)=1:X(5)=1
470 GOTO 430
480 PRINT CHR$(147)"MEDIA ARITMETICA DI ";:W=0
490 FOR I=1 TO DI-1:PRINT "(" Y(I) ")+";:W=W+Y(I):NEXT
500 PRINT CHR$(157)"=";
510 PRINT W/(I-1):GOSUB800:RETURN
520 PRINT CHR$(147)"INVIO DATI":PRINT
530 PRINT "SCEGLI:":PRINT
540 PRINT "1: REGISTRATORE"
550 PRINT "2: DRIVE"
560 PRINT "3: STAMPANTE"
570 PRINT: PRINT"4: RITORNO AL MENU"
580 GOSUB 800: IF A<1 OR A>3 THEN RETURN
590 IF A$="1"THEN 630
600 IF A$="2"THEN 680
610 IF A$="3"THEN 730
620 GOTO 520
630 INPUT"NOME DEL FILE";NO$:IF LEN(NO$)=0 THEN 520
640 OPEN 1,1,1,NO$
650 PRINT#1,ND:REM DIMENSIONE VETTORE
660 PRINT#1,DI:REM ULTIMO DATO
670 FOR I=1 TO DI-1:PRINT#1,Y(I):NEXT:CLOSE 1:RETURN
680 INPUT"NOME DEL FILE";NO$:IF LEN(NO$)=0 THEN 520
690 OPEN 8,8,8,NO$+",S,W"
700 PRINT#8,ND:REM DIMENSIONE VETTORE
710 PRINT#8,DI:REM ULTIMO DATO
720 FOR I=1 TO DI-1:PRINT#8,Y(I):NEXT:CLOSE 8:RETURN
730 PRINT CHR$(147)"SCEGLI:":PRINT
740 PRINT "1- DI SEGUITO"
750 PRINT "2- IN COLONNA"
760 GOSUB 800
770 OPEN 4,4:W=0
780 FOR I=1 TO DI-1:PRINT#4,Y(I);:IF A<>1 THEN PRINT#4
790 W=W+Y(I):NEXT:PRINT#4:PRINT#4,"MEDIA:"W:CLOSE4:RETURN
800 GET A$:IF A$="" THEN 800
810 A=VAL(A$):RETURN
820 END
```

L'ASSEMBLY, QUESTO SCONOSCIUTO

Per chi non si accontenta più del Basic, può essere utile andare a sciacquare i propri panni in Assembly

di **Domenico Pavone**

Prende il via da questo numero una serie di articoli finalizzati all'apprendimento della programmazione in linguaggio macchina, troppo spesso considerato un argomento misterioso e di impossibile decifrazione o riservato solo all'esclusiva setta dei programmatori "top" se non a quella imperscrutabile degli "hackers".

In realtà, superate le prime naturali difficoltà di adattamento all'oggetto computer, chiunque abbia già una certa dimestichezza con la programmazione in Basic può tranquillamente accostarsi a questa nuova materia. Una più approfondita conoscenza del Basic, d'altra parte, costituirà un vantaggio, grazie alla stessa impostazione logica dei due linguaggi nella progettazione di un programma, nella fase cioè posta a monte della sua stesura.

Come però verrà chiarito più avanti, usare il linguaggio macchina (l.m.) non vuol dire soltanto imparare l'uso di istruzioni che compiano un certo lavoro, ma anche (se non soprattutto) accostarsi maggiormente alla struttura "fisica" del calcolatore ed al suo modo di operare.

Non lasciatevi prendere dal panico: è tutto più semplice di quanto possa sembrare, soprattutto se affrontato con la dovuta gradualità.

Realizzare il videogame dell'anno dopo un paio di settimane di applicazione è sicuramente utopistico; ponendosi, invece, come obiettivo a breve scadenza la realizzazione di piccole routine in Assembler (che interagiscano con vostri programmi Basic), la possibilità di "scavare" a fondo dentro il C/64 (o C/128) ricompenserà ampiamente ogni sforzo, permettendovi procedure prima impensabili.

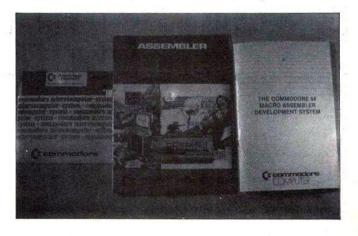
Se, poi, alla curiosità iniziale aggiungerete un po' di costanza...

Dal Basic al linguaggio macchina

Per capire che cosa sia realmente l'Assembler, è opportuno soffermarsi su alcune considerazioni di carattere generale, anche per sgombrare il campo da possibili equivoci e da frequenti luoghi comuni che circondano l'argomento.

Intanto cominciamo col precisare che Linguaggio Macchina e Assembler, contrariamente all'uso che spesso si fa dei due termini, nel loro stretto significato non sono affatto la stessa cosa. Vediamo di chiarirci le idee procedendo con ordine.

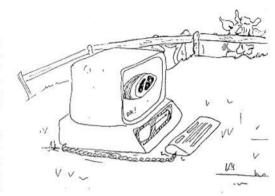
La prima necessità che si pone nella programmazione di un computer, è quella del dialogo con la macchina, ovvero il modo attraverso cui ottenere i risultati che vogliamo. Per usare un termine più complicato, è indispensabile una "interfaccia" logica, cioè un mezzo che consenta uno scambio di informazioni tra chi utilizza il computer ed il microprocessore che sta alla base di esso.

Il linguaggio macchina è più semplice di quanto comunemente si pensi

Chi conosce il Basic, anche in parte, può dedicarsi con soddisfazione al linguaggio macchina

Questo mezzo è, appunto, il "linguaggio".

Come certo saprete, esistono vari linguaggi, che in generale vengono suddivisi secondo una scala di livelli. In alto vi sono i cosiddetti linguaggi "evoluti", tra i quali troviamo il Basic, il Pascal, il Cobol e vari altri; in basso, all'ultimo gradino, c'è il linguaggio macchina.

Non si tratta, però, di una graduatoria di merito: il fatto che il Basic si trovi più in "alto" non significa che sia "migliore" del linguaggio macchina, ma solo che è più vicino al comune modo di esprimersi dell'uomo, e quindi di più facile comprensione.

Il linguaggio macchina, al contrario, segue le regole del microprocessore, che è in grado di "capire" solo alcune sequenze di numeri, come vedremo meglio in seguito; sarà quindi il programmatore, in questo caso, a doversi adattare alle esigenze del particolare tipo di microprocessore installato sul computer.

Per chiarire questa differenza, riferiamoci ad un esempio banale: supponendo che ci sia venuta improvvisamente fame, decidiamo di mangiare un panino imbottito. Possiamo prepararcelo da soli, oppure affidare l'incarico al nostro fedele maggiordomo inglese (di nome Basic, ovviamente).

Il Macro Assembler Commodore

Programmare in Linguaggio Macchina, senza uno strumento che ne semplifichi l'applicazione, è un'impresa decisamente ardua, soprattutto per chi si è appena accostato all'argomento.

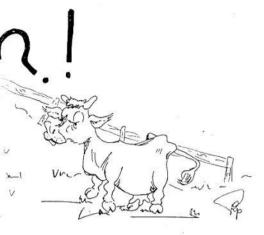
Per questa categoria di utenti, ma anche per chi ha già una certa conoscenza dell'Assembly, l'insieme dei programmi contenuti nella confezione originale del Macro Assembler Commodore rappresenta un aiuto più che valido.

Il dischetto di questa utility contiene infatti tutto ciò che è necessario per la stesura, l'assemblaggio e la revisione di routine scritte nel linguaggio più veloce che si possa implementare sul C/64.

Particolarmente comodo risulta l'editing, dotato della possibilità di utilizzare numeri di riga, proprio come nel Basic, ma con in più l'autonumerazione delle righe; si possono inoltre inserire nel testo dei commenti, ed è previsto l'uso di alcune direttive (non appartenenti allo standard dell'Assembly) per facilitare, nei programmi, l'inserimento di variabili, testi, eccetera.

Per le operazioni di debugging sono forniti ben due "monitor", con diversa installazione in memoria a seconda dell'area sulla quale si intende agire. Ogni opzione è implementata da programmi registrati in file separati, e poichè il dischetto è privo di qualsiasi tipo di protezione, è possibile trasferirli su altri dischetti di lavoro.

Sempre per lo stesso motivo, è possibile utilizzare differenti tipi di monitor (come il Supermon, o altri), qualora se ne possegga una versione più aggiornata o una alla quale si è più abituati.

Ultima notazione degna di nota è che il package è fornito con un buon manuale d'uso (in inglese), che comprende anche una mappa di memoria del C/64 e l'elenco completo delle istruzioni Assembly del microprocessore 6502.

Le fasi necessarie per portare a termine l'operazione possono essere così schematizzate:

1) Acquistare l'occorrente.
2) Affettare il pane.
3) Preparare gli ingredienti.
4) Metterli dentro il panino.
5) Richiudere il tutto.

Se si provvede da soli, sarà necessario seguire l'intera sequenza, fino all'ultimo punto, suddividendo, addirittura, ogni fase nelle sue componenti. Per esempio, la seconda fase comprenderà anche: aprire il cassetto delle posate, prendere un coltello, eccetera.

Si raggiungerà, quindi, il massimo di dettaglio nell'esecuzione delle varie operazioni.

Affidandoci, invece, al maggiordomo, il nostro intervento si potrà fermare alla semplice espressione del desiderio di nutrirci; penserà lui, su nostra istruzione, a realizzare le fasi indicate.

Nel primo caso abbiamo seguito una procedura che può essere paragonata all'uso di un linguaggio di basso livello (come il linguaggio macchina); nel secondo abbiamo adoperato un linguaggio "evoluto" (come il Basic) che ci ha consentito di evitare un bel po' di lavoro.

I comandi Basic non sono altro che gruppi di istruzioni l.m. poste in successione

Un chip per tutti i gusti

Nonostante la diversità di prestazioni che caratterizza i vari modelli Commodore, all'interno di tutti (Amiga e PC esclusi) batte lo stesso "cuore", un microprocessore della serie 6500.

Tale sigla, col trascorrere del tempo, è diventata 6502 nei vecchi VIC/20, 6510 nel C/64, 7510 nel C/16 ed infine 8502 nel C/128. La sua gestione interna, e quindi il suo linguaggio, rimangono tuttavia pressochè immutati, come pure la quantità massima di memoria gestibile.

A fare la differenza, talvolta anche grande, è l'apparato di "sostegno" in cui il microprocessore, ovvero la CPU (Central Process Unit), è inserito.

Tale apparato è costituito dalla specifica struttura hardware e dal cosiddetto "firmware", ossia il software residente nella memoria ROM del sistema che, nel C/64, è rappresentato dal Sistema Operativo e dall'interprete Basic.

Potenziando questi ultimi elementi, o aumentando la memoria gestibile dall'utente, si migliorano le capacità generali di un computer, ma poggiando sempre sulle stesse caratteristiche del microprocessore. Un eccesso in tal senso, pur portando certi vantaggi, può quindi "sovraccaricare" il sistema, per cui, ad esempio, pur disponendo di grosse quantità di memoria e di un maggior numero di comandi Basic, si avrà come conseguenza un eccessivo rallentamento delle operazioni.

Anche nello sfruttamento del 6502 si può quindi affermare che "in medio stat virtus", ed al C/64 non è certo la virtù che manca.

Una elevata velocità operativa richiede il l.m.

Si badi, però, che le operazioni necessarie per raggiungere lo scopo restano sempre le stesse; cambia solo il punto dal quale non sarà più indispensabile un nostro intervento diretto. Questa precisazione si impone per capire un'altra caratteristica dei due tipi di linguaggio: la differente velocità di esecuzione.

Semplicità o velocità?

Quanto detto fino ad ora può sembrare che deponga nettamente in favore del Basic; in raltà, a certi innegabili vantaggi legati ad una sua relativa facilità d'impiego, si accompagna una "lentezza" che ne rende l'utilizzo decisamente improponibile in molte applicazioni.

Come abbiamo accennato, infatti, il microprocessore del computer può comprendere un ordine solo se espresso nel "suo" linguaggio, fatto di numeri e di istruzioni assai elementari.

.Per eseguire qualunque comando più evoluto, sarà quindi necessario un "interprete" che faccia da tramite tra i due linguaggi, scomponendo ogni comando nelle sue più elementari istruzioni e provvedendo ad eseguirle nel linguaggio proprio del microprocessore. Tutto questo, come è facile intuire, porta ad un inevitabile rallentamento delle operazioni, aggravato ulteriormente da tutti i controlli necessari al Basic per funzionare correttamente (presenza di errori nella digitazione dei comandi, attuale posizione del cursore, eccetera).

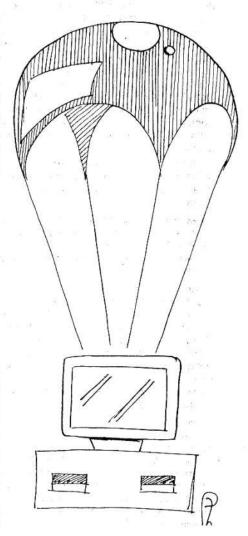
Tornaniamo all'esempio precedente: se facciamo preparare il panino dal maggiordomo risparmieremo, sì, del lavoro, ma occorrerà un tempo maggiore: dovremo infatti spiegargli (per giunta in inglese!) cosa vogliamo come ripieno, in che quantità, dove abbiamo riposto gli ingredienti, e magari attendere che abbia prima sbrigato qualche altra faccenda.

```
1 REM RIEMPE SCHERMO
2 REM C/64 BASIC
3 :
10 FOR X=1024 TO 2023
20 POKE X,81:NEXT
30 GOSUB 100
40 FOR X=1024 TO 2023
50 POKE X,32: NEXT: END
100 GET A$:IF A$="" THEN 100
110 RETURN
```

Vediamo però di verificare più in concreto tutto ciò, lasciando al computer il compito di mostrarci il salto di qualità che si opera con l'uso della sua... lingua madre.

Prima di proseguire occorre precisare che tutti gli esempi pratici che da ora in avanti verranno adottati, si riferiscono all'uso del C/64. Le stesse regole di programmazione, in generale, sono valide anche per gli altri modelli Commodore basati sul medesimo tipo di microprocessore (vedi riquadro).

In pratica, però, poichè il linguaggio macchina interagisce strettamente con l'architettura interna del calcolatore, e poichè questa cambia da modello a modello, lo stesso programma è difficilmente "traspor-

tabile" da un computer all'altro. La cosa risulterà più evidente in seguito, quando cominceremo a muoverci con più consapevolezza all'interno della memoria del computer. Per adesso accettatelo come un dato di fatto.

Ma torniamo all'argomento principale, e vediamo di toccare con mano la diversità di comportamento del nostro C/64 di fronte ad uno stesso compito, a seconda che si proceda in Basic oppure in l.m.

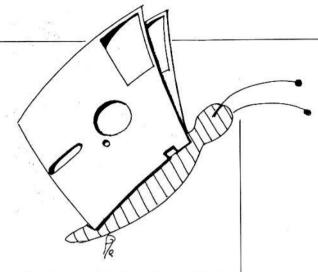
Cominciamo col digitare, e salvare su disco (o nastro), le poche righe del listato n.1. Il programma si limita a riempire lo schermo con il simbolo semigrafico del pallino pieno che si ottiene digitando Shift + "Q", il cui codice di schermo è 81.

Come saprete, inserendo tale codice tramite il comando Poke nelle locazioni da 1024 a 2023, se ne otterrà la visualizzazione sul video. Fatto ciò (righe 10 e 20), tramite il Gosub di linea 30 (e relativo ciclo di attesa di riga 100) si attende la pressione di un tasto qualsiasi, qindi si vuota lo schermo ripetendo l'operazione come sopra, ma stavolta usando il numero di codice 32, cioè lo spazio.

Impartendo il Run, si può notare come lo schermo vada progressivamente riempendosi e, dopo avere premuto un tasto, vuotandosi.

Ora, purchè si sia già provveduto a salvarlo, cancelliamo dalla memoria il programma con New (e Return) e copiamo il listato 2, badando a non commettere errori nella digitazione della sequenza di numeri.

Come si può notare, la struttura di questo secondo listato è notevolmente diversa dal precedente; l'unica funzione svolta dal programma è infatti la lettura, tramite Read...Data, di una serie di valori per poi "pokarli" in una certa area della memoria del computer. Questi valori non sono altro che i codici di un programma in linguaggio macchina, il quale svolge la stessa funzione del precedente in Basic.

Senza entrare, per adesso, nei particolari, lanciamo il programma con il consueto Run. Apparentemente non è successo nulla, ma in effetti ora è presente in memoria una routine l.m. che, per funzionare, attende che si impartisca (direttamente o da programma) un comando "Sys" seguito dal numero della locazione da cui essa inizia, in questo caso 49152.

Ora diamo un altro New. Con questo comando viene cancellato il programma Basic presente in memoria, mentre lo stesso non ha alcun effetto sulle routine in l.m. presenti (o, più correttamente, "allocate") in memoria.

A questo punto potremmo anche digitare, direttamente, Sys 49152, ma risulterà tutto più chiaro procedendo con maggiore "eleganza": senza spegnere l'apparecchio, ricarichiamo il primo programma (listato 1) e cancelliamo il comando End di riga 50; quindi digitiamo il listato 3, le cui linee di istruzioni si fonderanno con le precedenti.

Dando ora il Run, e ricordando di premere un tasto dopo ogni riempimento e svuotamento dello schermo, potrete constatare da soli l'abissale differenza che corre tra il modo di operare del Basic e quello del linguaggio macchina.

Apriamo una breve parentesi: i comandi della riga 90 fanno eseguire alla nostra routine l.m. il compito di stampare 1000 spazi vuoti; in effetti, però, lo stesso risultato può essere raggiunto modificando la linea nel modo seguente:

```
90 Print CHR$(147):End
```

In questo caso si è usato il Basic, eppure la velocità di esecuzione è pari a quella ottenuta con la nostra routine. Com'è possibile? Semplice: con l'istruzione Print CHR$(147) il C/64 attiva un "suo" programma in linguaggio macchina, che, ovviamente, esiste già nella memoria Rom, montata dal fabbricante nel calcolatore. Risulta talvolta inutile, quindi, ricorrere a programmi "esterni", purchè si sappia come attivare quelli già presenti nella ROM (memoria a sola lettura) del sistema. Nel caso in questione, per esempio, la riga suddetta può anche essere sostituita da:

```
90 Sys 65409: End
```

Osservare l'esecuzione di un compito a velocità elevatissima ripaga del modesto sforzo da compiere per operare in l.m.

Grazie ai potenti linguaggi-utility oggi disponibili, programmare in l.m. è semplice quasi come operare in Basic

Al di là dell'esempio citato, come avremo occasione di vedere, esistono moltissimi casi in cui è possibile sfruttare le notevoli capacità di queste routine "interne", molto spesso non accessibili per un linguaggio evoluto come il Basic.

Assembly, Assembler e Macroassembler

Come avete potuto constatare digitando il listato 2, il linguaggio macchina vero e proprio è composto esclusivamente da numeri, che identificano sia dati che istruzioni. Se a questo si aggiunge che tali numeri andrebbero manipolati secondo il sistema di numerazione binario, ce n'è abbastanza per scoraggiare chiunque.

Proprio per superare tali difficoltà, è stato creato uno strumento che semplifica il lavoro del programmatore, e questo è... l'Assembler.

Se ci rifacciamo all'ipotetica scala che classifica i linguaggi in base al loro livello di "evoluzione", nel gradino immediatamente superiore a quello del l.m. troviamo un altro linguaggio, l'Assembly, più comunemente definito "Assembler Simbolico".

Anch'esso è di basso livello, in grado cioè di eseguire solo le stesse operazioni elementari che caratterizzano il l.m, ma a differenza di quest'ultimo gestisce codici "mnemonici" che identificano le istruzioni. I codici di cui parliamo sono, in pratica, delle abbreviazioni di termini inglesi, ma in ogni caso ben più facili da ricordare che dei semplici numeri. Tanto per fare un esempio, l'equivalente dell'istruzione in linguaggio macchina 169, indicante il caricamento di un particolare registro (chiamato Accumulatore), in Assembly sarà LDA, che è l'abbreviazione di LoaD Accumulator.

Questi comandi, in ogni caso, dovranno poi essere tradotti negli equivalenti codici numerici comprensibili al microprocessore, compito che viene svolto da programmi cosiddetti "assemblatori", traduzione del vocabolo inglese "assembler", da cui l'uso del termine per indicare il linguaggio.

Avremo modo comunque, man mano che ci addentreremo nei "meandri" di questo linguaggio, di prendere confidenza con la sua terminologia. A questo punto, se ci è ben chiara la differenza esistente tra Basic,

```
1 REM RIEMPE SCHERMO
2 REM DEL C/64 (L.M.)
3 :
10 FOR X=49152 TO 49176
20 READ Y:POKE X,Y: W=W+Y
22 NEXT: IF W<>3910 THEN 90
25 :
30 DATA 169,000,133,251,169,004
40 DATA 133,252,169,081,162,004
50 DATA 160,000,145,251,136,208
60 DATA 251,230,252,202,208
70 DATA 244,096
80 END
90 PRINT"ERRORE DI DIGITAZIONE"
100 END
```

l.m. ed Assembler, si è già compiuto un grosso passo avanti nell'approccio alla programmazione in Assembly.

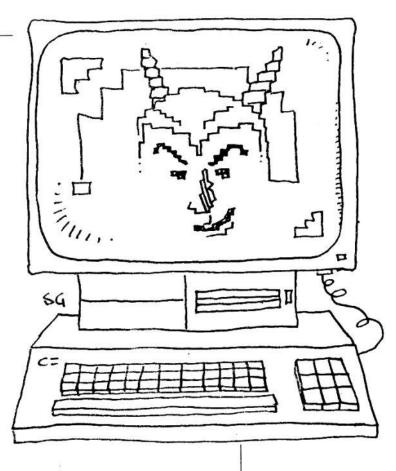
Per concludere questa prima escursione al di là del Basic, vediamo ora di quali strumenti concreti è necessario munirsi per sfruttare le potenzialità dell'Assembly. Com'è ovvio, non è possibile digitare direttamente un programma adoperando i simboli propri di questo linguaggio: l'interprete del Basic, operativo normalmente sin dal momento dell'accensione, rifiuterebbe con un "Syntax Error" qualunque istruzione a lui "estranea". Per ovviare a ciò, esistono particolari programmi che consentono di usare direttamente i codici mnemonici dell'Assembly. In generale, questi programmi possono essere divisi in due categorie: i cosiddetti "Monitor" e gli "Editor/Assembler".

I primi, pur consentendo la stesura ed il contemporaneo assemblaggio (= traduzione in codici numerici e installazione in memoria degli stessi) di routine, risultano tuttavia piuttosto scomodi da usarsi per redigere programmi di una certa consistenza.

In questo caso, infatti, è opportuno affidarsi ad un più completo e sofisticato editor/assembler. Quest'ultimo, tra l'altro, di solito è anche provvisto di un monitor, utile soprattutto nella fase di "debugging" (ricerca degli errori) per le sue capacità di esaminare e modificare rapidamente la memoria del computer. Il principale vantaggio di un Editor/Assembler consiste soprattutto nelle svariate opzioni di editing, che consentono una stesura dei programmi in Assembly estremamente facilitata, paragonabile a quella cui si è abituati con il Basic o nell'uso di un word processor.

Le specifiche caratteristiche di questa utility cambiano però in rapporto alla versione utilizzata. Sul mercato sono reperibili, infatti, vari Editor/Assembler, tanto su cartuccia che su disco (su nastro, purtroppo, c'è ben poco di realmente valido). Citiamo, tra gli altri, il programma EDNA che, alla comodità legata al tipo di supporto, contrappone un prezzo non proprio irrisorio, nonchè una certa difficoltà di reperimento.

Tra i prodotti su dischetto, troviamo un parto di mamma Commodore, il famoso economico e reperibilissimo "Macro Assembler".

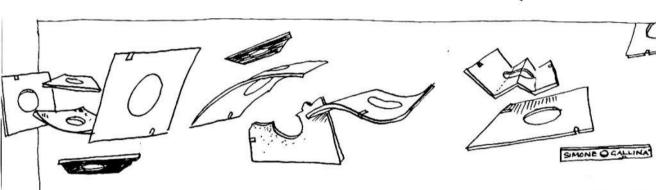
Considerato uno dei "classici", il package (vedi riquadro) è di estrema facilità e flessibilità di uso e possiede la non trascurabile

dote di essere distribuito dalla stessa Commodore. Questo significa che lo si può trovare in ogni punto di vendita Commodore, cioè dappertutto in Italia. Grazie a tali caratteristiche, e poichè per intendersi bene è necessario un certo "standard", nel trattare i prossimi argomenti che riguardano il linguaggio macchina faremo largo uso dei programmi (non protetti!) contenuti nel dischetto del Macro Assembler, chiarendone, nella pratica, le modalità di utilizzo.

```
60 GOSUB 100
70 POKE 49161,81:SYS 49152
80 GOSUB 100
90 POKE 49161,32:SYS 49152
92 END
```

In attesa dei prossimi articoli, quindi, vi consigliamo di procurarvi il package citato. Chi possiede il solo registratore (ma che aspettate a comprare un drive, magari compatibile?) stia pure tranquillo: potrà comunque seguire la trattazione sul linguaggio macchina.

LA TERZA VIA PER LEGGERE LA DIRECTORY DI UN DISCO

Vi sono molti sistemi per leggere una directory senza cancellare il programma presente nella memoria del C/64. Uno di questi...

di **Domenico Pavone**

Capita spesso, nella stesura di un programma, di trovarsi di fronte all'esigenza di inserire una routine per accedere alla directory di un dischetto. Nel caso più frequente ciò accade solo per esaminarne il contenuto, ma sovente anche per l'archiviazione dei file, la stampa su carta, eccetera.

E capita ancora più spesso, a chi programma in basic, di arrangiarsi come può, data l'esasperante lentezza dei due principali metodi per ottenere lo scopo desiderato. Senza entrare nel merito (basterebbe rileggere i numeri 22, 30 e 36 di C.C.C.), vi ricordiamo che le due fasi più seguite sono le seguenti:

- Aprire in lettura un file PRG col nome "$" e leggerlo tramite innumerevoli GET relative al drive.
- Leggere, tramite i comandi di accesso diretto, la traccia 18.

Il secondo sistema è leggermente più rapido del primo, ma siamo sempre ben lontani dall'ordine dei pochi secondi. Usando invece una routine in linguaggio macchina il problema cessa di esistere, ma per chi ancora non è del tutto addentro alle segrete cose dell'Assembly non è proprio come bere un bicchier d'acqua.

E' possibile però adottare un'altra tecnica, di estrema semplicità e notevolmente più veloce delle due menzionate. Non si tratta altro che di far eseguire alla macchina un semplice LOAD "$",8 caricando però la directory in un'area RAM lontana da quella in cui è scritto il nostro programma.

Una volta in memoria, disporremo, a partire da una locazione nota, di tutti i dati del dischetto, ordinati con regolarità. Sarà facile a questo punto qualunque operazione di lettura e manipolazione dei dati.

Una delle carenze del C/64 è quella di non avere un comando che consenta di visualizzare la Directory

Dal dire al fare

Per ottenere tutto questo, l'unico aiuto non strettamente basic possiamo farcelo dare dalla routine LOAD del Kernal. Chi non mastica il linguaggio macchina (per gli altri è sufficiente dare un'occhiata al disassemblato commentato) basterà sapere che, "chiamando" questa routine, è possibile caricare un file all'indi-

Anche per la gestione delle periferiche è meglio, ove possibile, ricorrere al L.M.

rizzo desiderato, purchè si usino precisi parametri.

Riferiamoci ora al listato 1, che permette di visualizzare la directory senza, ovviamente, perdere il programma presente in memoria.

Prima di tutto viene caricata la routine LM di LOAD (mediante la solita procedura Read... Data), installata a partire da 49152 (riga 40); potete usare anche un indirizzo della normale RAM basic, ma in questo caso dovete provvedere a sistemare i puntatori al tetto del basic (55-56) e all'inizio delle variabili stringa (51-52) operazione, questa, semplice solo per chi conosce perfettamente il basic; gli altri digitino il listato così com'è.

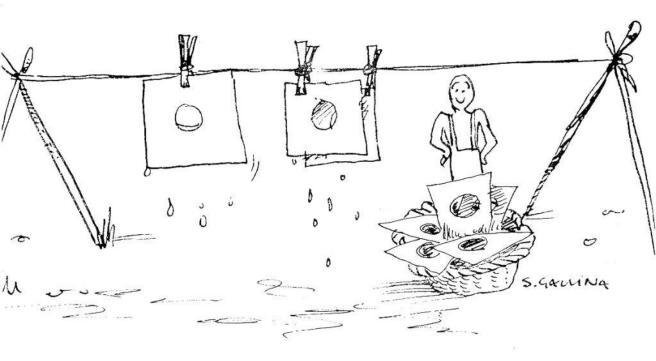
La routine LM carica la directory a partire da 49408 ($ C100) e anche qui, se ne siete in grado, potete intervenire per modificarne l'indirizzo, cambiando le linee Data 430 e 450 che ne rappresentano il byte basso e quello alto, nonchè il valore della variabile X alla riga 30.

In coda alla routine c'è il valore ASCII del dollaro ($), la cui posizione deve essere precisata nei Data di riga 390 e 410 sempre in formato basso/alto; una possibile variante può essere togliere del tutto la linea 470 (cambiando da 28 a 27 le Read in riga 40) e porre nei Data che puntano a "$", vale a dire 96 (basso) e 163 (alto), cioè l'indirizzo ROM 41824 che già contiene il valore 36, e che andrà bene in qualunque circostanza.

Una volta che la directory si trova in memoria, dopo l'attivazione della sys di riga 50, non resta che eseguire dei puntamenti alle posizioni in cui si trovano i dati che interessano, sommando a X (locazione d'inizio del caricamento) la posizione che si vuole leggere... ed il gioco è fatto.

Per sapere se si è giunti alla fine dell'elenco, basta controllare (riga 320) se alla quinta posizione dopo l'ultimo dato è presente la B (=66) di Blocks free, nel qual caso il programma si conclude.

Con il listato 2 si può apprezzare un uso più sofisticato dello stesso algoritmo, che consente di operare una scelta ed una eventuale modifica dei dati della directory. Dati che, assegnati ad un vettore, vengono stampati su carta in una forma esteticamente valida e pratica per incollarla sulla custodia del dischetto.

A parte, però, le righe che riguardano la stampa (470-540), è intuitivo che le applicazioni possibili dipendono solo dalle vostre necessità.

Disassemblato load directory

```
c000   lda #$00
c002   ldx #$08    ;apre un file sulla periferica n.8
c004   ldy #$00    ;con caricamento rilocato
c006   jsr $ffba   ;routine kernal 'setlfs' (decimale = 65466)
c009   lda #$01    ;lunghezza del nome del file
c00b   ldx #$1c    ;byte basso e
c00d   ldy #$c0    ;byte alto della posizione del nome del file
c00f   jsr $ffbd   ;routine kernal 'setnam' (decimale = 65469)
c012   lda #$00    ;operazione di caricamento (se 1=verify)
c014   ldx #$00    ;byte basso e
c016   ldy #$c1    ;byte alto indirizzo da cui iniziare il caricamento
c018   jsr $ffd5   ;routine kernal 'load' (decimale = 65493)
c01b   rts         ;ritorno al basic
c01c   24          ;nome file da caricare (decimale = 36 = $)
```

SCHEMA 1

TESTATA DEL DISCHETTO	
1 - 2	Puntano al prossimo blocco di dati
3 - 4	00
5	$12 (dec.18) - REVERSE ON
6	$22 (dec.34) - VIRGOLETTE
7 - 22	Nome disco + eventuali spazi ($20-dec.32)
23	Virgolette
24	Spazio
25 - 26	Caratteri della ID
27	Spazio
28 - 29	Caratteri 2A
30	00 (fine blocco dati)

SCHEMA 2

FILE DELLA DIRECTORY (con meno di 10 blocchi)	
1 - 2	Puntano al prossimo blocco di dati
3 - 4	Blocchi del file (low-hi)
5 - 6 - 7	Spazio ($20-dec.32)
8	Virgolette ($22-dec.34)
9 - 26	Nome file + virgolette + eventuali spazi
27 - 29	Caratterei del tipo di file
30 - 31	Spazio
32	00 (fine blocco dati)

SIMONE GALLINA

La directory viene gestita come se si trattasse di un programma Basic

```
10 PRINTCHR$(147):POKE 53280,0
   :POKE 53281,0:PRINTCHR$(155
   )
20 PRINTSPC(250)"LETTURA DIREC
   TORY..."
30 X=49408:
40 FOR I=0 TO 28:READ A:POKE 4
   9152+I,A:NEXT
50 SYS49152
70 REM    **** LEGGE NOME E ID
80 :
90 FOR I=X+4 TO X+25:ND$=ND$+C
   HR$(PEEK(I)):NEXT
100 PRINTCHR$(147):PRINTSPC(3)N
    D$:PRINT
115 X=X+30
120 NF$="":TF$=""
140 :
150 REM   **** BLOCCHI DEL PROG
160 :
170 BL=PEEK(X+2)+PEEK(X+3)*256
210 REM   **** NOME E TIPO DI FI
    LE
230 FOR I=X+5 TO X+31:NF$=NF$+C
    HR$(PEEK(I)):NEXT
280 PRINTBL;NF$
300 REM   **** FINE DELLA DIRECT
    ORY?
320 IF PEEK(X+36)=66 THEN 335
330 X=X+32:GOTO 120
335 PRINTPEEK(X+34)+PEEK(X+35)*
    256"BLOCCHI LIBERI":END
340 :
350 REM   **** LOAD DIRECTORY
360 :
370 DATA  169,0,162,8,160,0,32,
    186,255
380 DATA  169,1,162
390 DATA  28:REM * BYTE BASSO P
    OSIZ.DI '$'
400 DATA  160
410 DATA  192:REM * BYTE ALTO P
    OSIZ.DI '$'
420 DATA  32,189,255,169,0,162
430 DATA  0:REM  BYTE BASSO POS
    . DIRECTORY
440 DATA  160
450 DATA  193:REM  BYTE ALTO PO
    S.DIRECTORY
460 DATA  32,213,255,96
470 DATA  36:REM  ******* ASCII
    DI '$'
```

COME E' STRUTTURATA LA DIRECTORY

Se andate a curiosare con un qualsiasi programma di monitor, a partire dalla locazione di inizio basic ($0801 = 2049 decimale), vi accorgerete che i dati della lista del dischetto sono memorizzati come se fossero comuni linee basic, aventi come numero di riga il numero di blocchi dei files. Il nome del dischetto ha sempre numero di linea 0, ed è quindi il primo ad essere visualizzato dopo il rituale LIST.

A riprova di ciò il comando RUN, in seguito al quale appare la segnalazione di Syntax Error in 0, conferma che ciò che il computer "vede", al posto dell'intestazione del dischetto, è una qualsiasi istruzione basic, naturalmente... errata.

L'intestazione del dischetto occupa le prime 30 locazioni.

Dalla trentunesima in poi si susseguono i dati riguardanti i files, ognuno dei quali occupa un blocco di 32 locazioni.

Per ciò che riguarda la disposizone delle prima trenta locazioni, potete fare riferimento alla figura 1, mentre in figura 2 è mostrato il modo in cui viene disposto in memoria, posizione per posizione, un blocco dati riguardante un file con meno di dieci blocchi.

Quest'ultima precisazione si rende necessaria perchè esistono alcune differenze a seconda che il numero di blocchi di un file sia compreso tra 1 e 9, tra 10 e 99 oppure tra 100 e 664, in pratica a seconda che la cifra che li specifica sia composta di 1, 2 oppure 3 caratteri.

Tale differenza riguarda le posizioni 5, 6, 7, 30 e 31. Come notate dallo schema di figura 2, le locazioni sono occupate da 5 spazi, ma il loro rapporto può variare. Se il numero di blocchi è minore di dieci, avremo una disposizione come in figura; se si superano i 9 allora avremo solo due spazi in posizione 5 e 6, mentre in coda a nome e tipo di file gli spazi diventeranno 3.

Analogamente, se i blocchi sono più di 99, troveremo solo uno zero in posizione 5 e da 28 in poi ne troveremo 4. Questo modo di organizzare le cose, come potete intuire, è stato studiato per una visualizzazione ordinata della directory sullo schermo.

L'ultimo blocco di dati contiene il numero di blocchi liberi (in terza e quarta posizione), e dalla quinta alla quindicesima il messaggio "Blocks free." più alcuni spazi.

```
10 PRINTCHR$(147):POKE 53280,0
   :POKE 53281,0:PRINTCHR$(155
   )
20 FOR I=1 TO 52:LN$=LN$+CHR$(
   192):NEXT
30 FOR I=1 TO 13:SP$=SP$+CHR$(
   32):NEXT:DIM  PF$(144)
35 PRINTSPC(250)"LETTURA DIREC
   TORY..."
40 REM  LETTURA DIRECTORY
50 X=49408
60 FOR I=0 TO 28:READ A:POKE 4
   9152+I,A:NEXT
70 SYS49152
80 FOR I=X+6 TO X+25:IF PEEK(I
   )=34 THEN ND$=ND$+CHR$(32):
   GOTO 100
90 ND$=ND$+CHR$(PEEK(I))
100 NEXT:X=X+30:PRINTCHR$(147)
110 BL$="":NF$="":TF$=""
120 Y1=X+8:Y2=X+26:Y3=X+36
130 BL=PEEK(X+2)+PEEK(X+3)*256:
    BL$=STR$(BL)
140 IF LEN(BL$)<5 THEN BL$=BL$+
    CHR$(32):GOTO 140:REM --- A
    GGIUNGE SPAZI
150 IF BL>9 THEN Y1=Y1-1:Y2=Y2-
    1
160 IF BL>99 THEN Y1=Y1-1:Y2=Y2
    -1
170 FOR I=Y1 TO Y1+15:IF (PEEK(
    I))=34 THEN NF$=NF$+CHR$(32
    ):GOTO 190
180 NF$=NF$+CHR$(PEEK(I))
190 NEXT
200 FOR I=Y2 TO Y2+2:TF$=TF$+CH
    R$(PEEK(I)):NEXT
210 REM  SCELTA E MODIFICA FILE
    S
220 PRINTBL$+NF$+CHR$(32)+TF$;:
    PRINT TAB(26)"DA STAMPARE?"
    ;CHR$(145)
230 W$="":GET W$:IF W$="" THEN
    230
240 IF W$="N" THEN PRINT TAB(26
    )SP$:GOTO 440
250 IF W$="S" THEN J=J+1:GOTO 2
    70:REM --- ASSEGNA INDICE
260 GOTO 230
270 PRINT TAB(26)" MODIFICHE?
    ";CHR$(145)
280 W$="":GET W$:IF W$="" THEN
    280
290 IF W$="N" THEN PRINT TAB(26
    );SP$:GOTO 410
300 IF W$="S" THEN PRINT TAB(26
    );SP$:GOTO 320
310 GOTO 280
320 PRINTCHR$(145)CHR$(18);"NOM
    E";CHR$(146);CHR$(32);NF$+S
    P$:PRINTCHR$(145); TAB(5);
330 OPEN 1,0:INPUT#1,EF$:CLOSE
    1:IF LEN(EF$)>16 THEN PRINT
    :GOTO 320
340 NF$=EF$:PRINT
350 IF LEN(NF$)<16 THEN NF$=NF$
    +CHR$(32):GOTO 350
360 PRINTCHR$(145)CHR$(18);"BLO
    CCHI";CHR$(146);BL$+SP$:PRI
    NTCHR$(145); TAB(8);
370 OPEN 1,0:INPUT#1,EB$:CLOSE
    1:IF LEN(EB$)>3 OR VAL(EB$)
    >664 THEN PRINT:GOTO 360
380 BL$=CHR$(32)+EB$:PRINT
390 IF LEN(BL$)<5 THEN BL$=BL$+
    CHR$(32):GOTO 390
400 REM  VETTORE DA STAMPARE  P
    F$(J)
410 PF$(J)=BL$+NF$+CHR$(32)+TF$
420 PRINTCHR$(145)CHR$(5)PF$(J)
    CHR$(155)
430 REM  ----------------
440 IF PEEK(Y3)=66 THEN BF=PEEK
    (Y3-2)+PEEK(Y3-1)*256:BF$=S
    TR$(BF):GOTO 470
450 X=X+32:GOTO 110
460 REM  STAMPA SU CARTA
470 PRINT:PRINT:PRINT TAB(12)"O
    K PER STAMPARE?"
480 GET A$:IF A$="" THEN 480
490 IF A$="N" THEN PRINTCHR$(14
    7):END
500 IF A$<>"S" AND A$<>"S" THEN
     480
510 PRINTCHR$(147)SPC(254)"STAM
    PA..."
520 OPEN 4,4:PRINT#4,LN$:PRINT#
    4,SPC(5)CHR$(14)ND$;CHR$(15
    ):PRINT#4,LN$
530 FOR I=1 TO J STEP 2:H=I+1:P
    RINT#4,PF$(I)+CHR$(32);:PRI
    NT#4,PF$(H):NEXT
540 PRINT#4,BF$+" BLOCCHI LIBER
    I":PRINT#4,LN$:PRINT#4:CLOS
    E 4:PRINTCHR$(147):END
550 REM  ----------------
560 DATA  169,0,162,8,160,0,32,
    186,255
570 DATA  169,1,162
580 DATA  28:REM * BYTE BASSO P
    OSIZ.DI '$'
590 DATA  160
600 DATA  192:REM * BYTE ALTO P
    OSIZ.DI '$'
610 DATA  32,189,255,169,0,162
620 DATA  0:REM  BYTE BASSO POS
     DIRECTORY
630 DATA  160
640 DATA  193:REM  BYTE ALTO PO
    S.DIRECTORY
650 DATA  32,213,255,96
660 DATA  36:REM  ASCII DI '$'
670 END
```

Le righe Data devono sempre essere copiate con estrema attenzione

REVERSE WINDOW SU 40 COLONNE

Come simulare, nel "modo" 40 colonne, una caratteristica peculiare del C/128 in 80 colonne

di **Domenico Pavone**

Nell'uso del C/128 in modo 80 colonne è possibile porre in reverse lo schermo premendo in successione i tasti ESC(ape) + "R", oppure, da programma, con Print Chr$(27)+"r".

Questa opzione viene a mancare se si lavora su 40 colonne, ed in ogni caso anche sulle 80 colonne funziona su tutto lo schermo, senza operare discriminazioni se si è impostata una finestra.

Per chiunque "mastichi" un po' di Assembly non costituisce certo un problema invertire la visualizzazione del video standard a 40 colonne, ma può risultare interessante far sì che venga posta in reverse solo la finestra attiva in quel momento, in modo da evidenziare certe elaborazioni di un nostro programma.

Ed è proprio il compito che svolge il breve listato di queste pagine, il quale, tra l'altro, fa uso di alcune routine di sistema la cui conoscenza può risultare utile a tutti, data la ormai cronica scarsità di documentazione sul C/128.

Come funziona

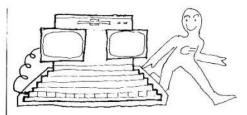
Come potete notare dal disassemblato (e relativo caricatore Basic), la routine è allocata a partire da $1300 (4864 decimale), un'area RAM libera da interferenze Basic, e va mandata in esecuzione con un banale:

BANK 15: SYS 4864

Se già vi "trovate" nel banco 15, potete ovviamente omettere la prima parte del comando. La SYS di attivazione funziona come un interruttore s/w bistabile tra il modo normale e quello reverse.

In altra parole, se impartite di nuovo SYS 4864 mentre la finestra è in reverse, questa tornerà allo stato normale. Come già accennato, verrà invertita la visualizzazione della window impostata al momento della chiamata alla subroutine; se non ne è stata definita alcuna, verrà considerato, come "finestra", l'intero schermo.

Non va dimenticato che viene modificato solo lo stato dello schermo, mentre rimane immutata l'impostazione di scrittura. Quindi, se dopo avere invertito l'area video vi si scrive sopra qualcosa, questo avverrà nel modo nor-

male, e non reverse, a meno che non lo specifichiate voi stessi nel vostro programma (o in modo diretto).

Inoltre, per un corretto uso della routine, occorre tenere presente che quando si usano i comandi di impostazione del colore (COLOR), occorre pulire lo schermo (SCNCLR oppure shift + clr/home) dopo la modifica del colore e non prima, o in reverse appariranno i colori precedenti.

Tanto per vedere un modo di utilizzazione di "Reverse window" potete copiare e mandare in esecuzione le poche righe del programma DEMO: a voi il compito di trovare qualcosa di più creativo.

Nei dettagli

Il semplice cambiamento di colore di una finestra produce un risultato di sicuro effetto

Vediamo ora, più in dettaglio, come opera il programma.

Per stabilire le esatte dimensioni della finestra video, si fa riferimento a quattro locazioni di pagina zero in cui il sistema operativo immagazzina i parametri come segue:

$E4 - riga inferiore
$E5 - riga superiore
$E6 - colonna sinistra
$E7 - colonna destra

Da tenere presente che i valori sono compresi tra 0 e 24 per le righe e tra 0 e 39 per le colonne.

Come primo passo, "Reverse window" pone nelle locazioni $16 e $17 l'indirizzo della prima cella di memoria video (40 colonne) che, come tutti sappiamo, è 1024 (in esadecimale $400).

Ovviamente tale indirizzo sarà depositato in formato basso/alto, per essere poi usato come puntatore per un indirizzamento indiretto. La scelta delle due locazioni non è casuale, come vedremo tra poco.

Facendo riferimento al disassemblato, da 1308 a 1314 si somma 40 all'indirizzo di inizio ($400) del video tante volte quante sono le righe contenute in $E4, in pratica si moltiplica Inizio * Riga superiore, ricorrendo alla subroutine di somma che inizia a 1343.

Aritmetica in L.M.

E qui apriamo una breve parentesi. Per eseguire operazioni aritmetiche in linguaggio macchina, come forse saprete, si usano due "contenitori", chiamati FAC1 e FAC2 (da "Floating ACcumulator": accumulatori in virgola mobile) nei quali devono essere depositati i valori da elaborare.

Senza entrare troppo nel merito, ai non esperti basti sapere che in essi i numeri sono rappresentati non, come di solito, da due bytes, ma in un formato particolare detto "floating point", per cui prima di ogni operazione è necessario convertire un numero nel suddetto formato, e alla fine occorrerà riconvertire il risultato in un numero a due byte. Questo lavoro, come pure l'operazione in senso stretto, viene svolto da alcune routine di sistema allocate nella ROM dell'interprete Basic, e di alcune di esse si serve appunto la subroutine di somma di "Reverse window".

La prima di cui si fa uso è allocata a partire da $793C (altro possibile ingresso è $AF03) e si limita a caricare FAC1 col valore contenuto nel registro Y (byte basso) e nell'Accumulatore (byte alto); il salto a $8C38 (o anche ad $AF6C) trasferisce FAC1 in FAC2.

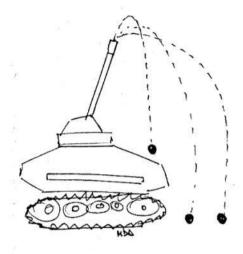
Nelle successive istruzioni della routine si carica ancora FAC1 con il valore decimale 40 (numero colonne) e col salto a $8848 (anche qui si può usare un altro ingresso a $AF1B) si effettuerà la somma tra FAC1 e FAC2, cioè Inizio + 40. Il risultato della somma si trova in FAC1, e da lì viene prelevato dalla routine di sistema locata a $8815 (anche a $AF0C), la quale lo trasforma in due bytes e lo deposita in $16 - $17 nel formato basso/alto.

A questo punto si torna al programma principale, che calcolerà il numero di righe e di colonne di cui è composta la finestra; questi valori serviranno come contatori per la effettiva operazione di reverse. Per queste ultime fasi, di comprensione piuttosto semplice, è sufficiente uno sguardo al commento del disassemblato.

Prima di concludere, una breve nota sulle routine di sistema preposte alle operazioni aritmetiche. Nel nostro disassemblato sono utilizzati i loro ingressi effettivi nell'area della Rom Basic, ma di essi esiste anche una tavola vettorizzata, cioè una serie di indirizzi che saltano alle varie routine, come di solito avviene per il kernal.

Tale accorgimento è in funzione di eventuali modifiche del Basic 7.0 da parte della Commodore. Per cui, se per caso (ammesso che ne esistano) disponete di una macchina "diversa", agli indirizzi usati in "Reverse window" potete sostituire quelli accennati tra parentesi nella descrizione appena ultimata.

```
100 REM REVERSE WINDOW C/128 40 COLONNE
110 REM ATTIVAZIONE = BANK 15:SYS 4864
120 :
130 BANK15:FAST
140 FORX=0TO90:READA:B=B+A
150 POKE4864+X,A:NEXT:SLOW
160 IFB<>12580THEN310
170 DATA 169,000,162,004,133,022,134
180 DATA 023,166,229,240,009,134,250
190 DATA 032,067,019,198,250,208,249
200 DATA 165,228,229,229,133,253,230
210 DATA 253,230,253,024,165,231,229
220 DATA 230,133,254,230,254,230,254
230 DATA 164,230,177,022,024,105,128
240 DATA 145,022,200,198,254,208,244
250 DATA 032,067,019,198,253,208,224
260 DATA 032,080,193,096,164,022,165
270 DATA 023,032,060,121,032,056,140
280 DATA 160,040,169,000,032,060,121
290 DATA 032,072,136,032,021,136,096
300 END
310 SCNCLR:PRINT"ERRORE NEI DATA"
320 SLOW:END
```

```
100 REM PRIMA DI ATTIVARE IL PROGRAMMA
110 REM CARICARE REVERSE WINDOW L.M.
120 :
130 PRINT"SS"
140 COLOR0,1:COLOR4,1:COLOR5,16:SCNCLR
150 REM 2 HOME PER USCIRE DALLA FINESTRA
160 PRINT"SCHERMO SENZA FINESTRA":LIST
170 GOSUB240
180 WINDOW5,10,30,20
190 PRINT"]RFINESTRA":LIST
200 GOSUB240:COLOR 0,3
210 WINDOW10,13,35,23:LIST
220 GOSUB240
230 PRINT"]PREMI DUE VOLTE CLR/HOME":END
240 BANK15:FORX=1TO3:SYS4864
250 SLEEP1:NEXT:RETURN
```

DISASSEMBLATO REVERSE WINDOW 128

```
---------------------------------------------------------
1300 LDA #$00       Alloca in $16-$17
1302 LDX #$04       indirizzo di INIZIO
1304 STA $16        dell'area video
1306 STX $17        ($0400) in forma low/hi.
---------------------------------------------------------
1308 LDX $E5        Riga superiore finestra.
130A BEQ $1315      Se riga = 0 salta a $1315.
130C STX $FA        Numero riga sup. in $FA.
130E JSR $1343      INIZIO = INIZIO + 40.
1311 DEC $FA        Riga = riga - 1.
1313 BNE $130E      Se riga <> 0 salta a 130E.
---------------------------------------------------------
1315 LDA $E4        Riga inferiore finestra
1317 SBC $E5        meno riga superiore.
1319 STA $FD        In $FD numero totale
131B INC $FD        righe della finestra.
131D INC $FD
---------------------------------------------------------
131F CLC
1320 LDA $E7        Col. inferiore finestra
1322 SBC $E6        meno col. superiore.
1324 STA $FE        In $FE numero totale
1326 INC $FE        colonne della finestra.
1328 INC $FE
---------------------------------------------------------
132A LDY $E6        In Y col. di inizio finestra.
132C LDA ($16),Y    Legge schermo da INIZIO + Y.
132E CLC            Inverte il carattere letto
132F ADC #$80       aggiungendo 180 (dec) e lo
1331 STA ($16),Y    riscrive sullo schermo.
---------------------------------------------------------
1333 INY            Y = Y + 1.
1334 DEC $FE        Col. finestra = col.- 1.
1336 BNE $132C      Se col.<> 0 salta a $132C.
1338 JSR $1343      INIZIO = INIZIO + 40.
133B DEC $FD        Righe finestra = righe - 1.
133D BNE $131F      Se righe <> 0 salta a $131F.
---------------------------------------------------------
133F JSR $C150      Porta il cursore a HOME.
1342 RTS            RETURN.
---------------------------------------------------------
1343 LDY $16        Preleva indirizzo
1345 LDA $17        di INIZIO e
1347 JSR $793C      lo carica in FAC1.
134A JSR $8C38      Trasferisce FAC1 in FAC2.
---------------------------------------------------------
134D LDY #$28       Setta Y e A con
134F LDA #$00       decimale 40 e
1351 JSR $793C      lo carica in FAC1.
1354 JSR $8848      Somma FAC1 + FAC2.
1357 JSR $8815      Trasferisce FAC1 in $16 - $17.
---------------------------------------------------------
135A RTS            RETURN
```

UN COMANDO IN PIU' PER IL MONITOR DEL C/128

Come manipolare la routine di interpretazione dei comandi L.M. per inserirne altri di nostra creazione

di **Domenico Pavone**

Se siete degli estimatori dell'Assembly, o se comunque non disdegnate qualche sbirciatina all'interno di programmi e subroutine non scritte in Basic, certamente avrete apprezzato la presenza nel C/128 di un monitor per il linguaggio macchina.

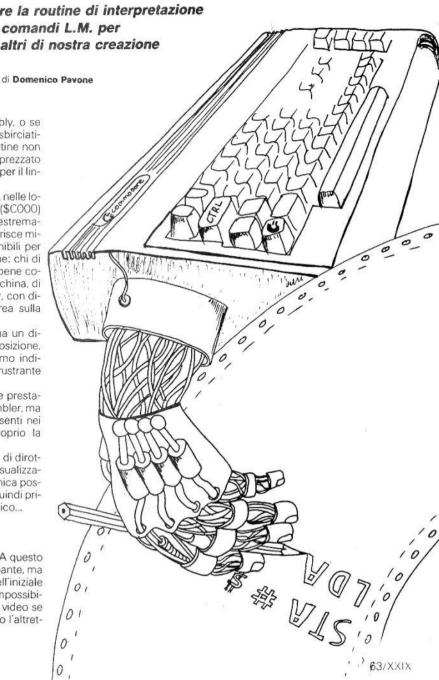
Il fatto che sia già residente in ROM, nelle locazioni da 45056 ($B000) a 49152 ($C000) di banco 14 e 15, ne rende l'utilizzo estremamente comodo, in quanto non interferisce minimamente con le aree RAM disponibili per collocarvi i programmi o le subroutine: chi di voi ne ha già usato uno sul C/64 sa bene come occorra disporre, per quella macchina, di svariate versioni dello stesso monitor, con diversa allocazione a seconda dell'area sulla quale si intende lavorare.

A questo vantaggio si accompagna un discreto corredo di comandi a disposizione, che, tuttavia, risulta limitato al minimo indispensabile e, quindi, spesso un po' frustrante per il programmatore.

Certo non si possono pretendere le prestazioni di un vero e proprio editor assembler, ma di alcune opzioni, normalmente presenti nei monitor "esterni", se ne sente proprio la mancanza.

Tanto per citarne una, la possibilità di dirottare verso la stampante la normale visualizzazione sul video. Per far ciò l'unica tecnica possibile è quella di effettuare da Basic, quindi prima di "entrare" in monitor, un classico...

OPEN 4,4:CMD 4

...e poi dare il comando MONITOR. A questo punto l'output sarà diretto alla stampante, ma senza potere eliminare la scrittura dell'iniziale stato dei registri e, soprattutto, nell'impossibilità di tornare alla visualizzazione su video se non uscendo dal monitor e digitando l'altrettanto classico PRINT# 4:CLOSE 4.

Il tasto Stop

Occorre prestare una certa attenzione nel caso si prema il tasto Run/Stop mentre, per esempio, si sta eseguendo la stampa su carta di un disassemblato. L'operazione verrà interrotta, ma si tornerà al modo video; in questo caso è indispensabile dare ugualmente un comando P per chiudere il file verso la stampante, anche se in effetti è stato riattivata l' uscita verso lo schermo.

Se, comunque, doveste dimenticarvene, non succederà niente di particolarmente grave: basterà, al successivo utilizzo del comando P, ripeterlo più di una volta ad una eventuale segnalazione di errore del kernal. Se però volete ovviare, potete aggiungere qualche istruzione che controlli il tasto stop (e si regoli di conseguenza) oppure una routine che segnali da qualche parte sullo schermo quale output è attivo (magari utilizzando una finestra o colorando il bordo), o qualunque alternativa vi venga in mente.

Gli sviluppi

Sulla base di quanto detto si può potenziare a volontà il monitor del C/128 e, perchè no, magari arrivare a progettare un vero e proprio editor per Assembly sfruttando come base quanto già mette a disposizione il nostro potente computer.

Sulla base di questi dati possiamo "costruire" un nuovo comando che risolva il problema dell'output su stampante, e che, in ogni caso, può servire come esempio per ulteriori e più sofisticate implementazoni. Seguendo il disassemblato della routine pubblicata (che può essere mandata in esecuzione dal relativo caricatore Basic), vediamone in dettaglio il funzionamento.

Come funziona "P"

Il breve programma, una volta lanciato, fa sì che il monitor riconosca come valido il comando P (lo stesso di altri tipi di monitor), che funzionerà come un interruttore: una volta impartito, l'output di qualunque comando che preveda una visualizzazione sullo schermo verrà diretto alla stampante.

Per tornare alla periferica di default, cioè il video, basterà di nuovo digitare P e premere Return.

Questo sistema non brilla certo per la comodità di uso, ma, come avrete pensato in molti, in mancanza d'altro...

Come agisce Monitor

A tale deficienza, e volendo anche ad altre, si può ovviare sfruttando una possibilità che viene offerta (intenzionalmente) dalla struttura di accesso ai comandi del monitor.

Per essere più chiari, vediamo, in sintesi, che cosa succede quando si dà il comando MONITOR.

Dopo alcune routine di inizializzazione, viene visualizzato lo stato dei registri, poi il sistema resta in attesa che venga digitato un comando.

Quando ne diamo uno, il suo valore ASCII viene immagazzinato nell'accumulatore (ricordiamo che tutti i comandi del monitor sono di un solo carattere) e quindi, per essere interpretato, il sistema operativo salterà ad una routine il cui indirizzo è contenuto in RAM alle locazioni 814 e 815 ($32E-$32F).

Come forse avrete già capito, se alteriamo il contenuto di queste locazioni in modo che puntino ad un nostro programma in linguaggio macchina, sarà possibile fare in modo che vengano accettati ed eseguiti altri comandi di nostra creazione.

```
2 REM ********************************
3 REM **    IMPLEMENTAZIONE COMANDO    **
4 REM **    PER OUTPUT  DEL MONITOR    **
5 REM **     DEL C128 SU STAMPANTE     **
6 REM ** ---------------------------- **
7 REM **      BY DOMENICO PAVONE       **
8 REM ********************************
9 :
100 SCNCLR:BANK 15
110 FOR X=0 TO 61:READ A: B=B+A
120 POKE 3072+X,A: NEXT
130 IF B<>7421 THEN PRINT"ERRORE! ":END
140 PRINT"COMANDO MONITOR [P] ATTIVATO"
150 SYS 3072
160 :
170 DATA 169,015,141,046,003,169,012
180 DATA 141,047,003,169,000,133,251
190 DATA 096,201,080,208,040,165,251
200 DATA 208,023,032,090,085,169,004
210 DATA 170,160,000,032,056,247,032
220 DATA 189,239,162,004,032,201,255
230 DATA 230,251,208,010,169,004,032
240 DATA 136,241,032,038,242,198,251
250 DATA 076,139,176,076,006,176
260 END
```

L'indirizzo contenuto nel vettore $32E - $32F, nel solito formato byte basso / byte alto, è normalmente $B006 (dec: 45062), che è l'ingresso alla sezione che controlla se un comando è corretto. In caso di errore, stampa un punto interrogativo e torna alla routine principale di accettazione ed esecuzione, allocata a partire da $B08B (dec: 45195).

La routine è allocata a partire da $C00 (dec: 3072), un'area normalmente riservata al buffer di input di una eventuale interfaccia RS-232; ciò è stato fatto per lasciare libera la più "consueta" area da $1300.

Se necessario, comunque, potete spostarla altrove, tramite il comando T dello stesso monitor, modificando solo il valore presente dopo le prime due istruzioni LDA: la prima operazione che compie il programma è infatti quella di installare nel vettore $32E - $32F l'indirizzo $C0F, ove sarà presente una routine per la gestione del nuovo comando. Oltre a modificare il contenuto di queste due locazioni, viene anche azzerata la cella di memoria $FB (dec: 251), che verrà utilizzata come flag per stabilire se il comando P è attivato oppure no.

Modificando il suddetto vettore, quando digiteremo un comando in ambiente monitor, il sistema salterà prima di tutto alla nostra routine, situata a $C0F, avendo nell'Accumulatore il valore ASCII della lettera che costituisce il comando stesso.

Questo viene comparato con $50 (dec: 80), per valutare se si tratta di una P: in caso negativo si salta alla normale gestione dei comandi ($B006), altrimenti si testa la locazione $FB (cioè il flag). Se questa conterrà zero, si aprirà in uscita un file con numero di periferica 4 (stampante) usando le adeguate routine del kernal (vedi commento nel disassemblato); si porrà quindi ad 1 la cella di memoria $FB e si salterà alla routine principale di attesa ed esecuzione dei comandi del monitor ($B08B).

Se invece $FB contiene un valore diverso da zero, allora si provvede a chiudere il file precedentemente aperto, si azzera $FB e si salta, anche in questo caso, a $B08B.

Con la tecnica descritta si potrà dirottare, verso la stampante, non solo un eventuale disassemblato, ma anche un dump di memoria o il semplice stato dei registri. Se dovessero verificarsi situazioni di errore, queste verranno segnalate dal sistema, ma nella forma usata dal kernal: un ERROR#2 significherà file già aperto, ERROR#3 file non aperto, ERROR#5 periferica non presente.

DISASSEMBLATO COMANDO MONITOR <P>

```
========================================
C00  LDA #$0F   Dirotta indirizzo
C02  STA $032E  contenuto nel
C05  LDA #$0C   vettore di esecuzione
C07  STA $032F  comandi monitor.
----------------------------------------
C0A  LDA #$00   Azzera locazione
C0C  STA $FB    $FB (dec: 251) e
C0E  RTS        RETURN.
========================================
C0F  CMP #$50   In A c'e' la lettera P?
C11  BNE $0C3B  Se no, salta a $B006.
C13  LDA $FB    Se $FB <> 0, salta a
C15  BNE $0C2E  routine chiusura file.
----------------------------------------
C17  JSR $555A  Esegue un PRINT.
C1A  LDA #$04   Setta file n. 4,
C1C  TAX        periferica n.4,
C1D  LDY #$00   ind.secondario 0, per
C1F  JSR $F738  routine kernal SETLFS.
C22  JSR $EFBD  Routine kernal OPEN.
C25  LDX #$04   File n.4 aperto in
C27  JSR $FFC9  uscita (kernal JCKOUT).
C2A  INC $FB    $FB = 1.
C2C  BNE $0C38  Salta a $B08B.
----------------------------------------
C2E  LDA #$04   Chiude file n.4 tramite
C30  JSR $F188  routine kernal CLOSE.
C33  JSR $F226  Resetta l'input/output.
C36  DEC $FB    $FB = 0.
----------------------------------------
C38  JMP $B08B  Routine principale di
                accettazione comandi.
----------------------------------------
C3B  JMP $B006  Routine normale di
                controllo comandi.
========================================
```

Stampare il disassemblato di un programma l.m. risulta praticamente indispensabile per chiunque

ANIMAZIONI ULTRA-RAPIDE PER IL C/128

E' possibile simulare con un C/128, anche se in minima parte, la veloce gestione grafica dell'Amiga?

di **Domenico Pavone**

Nel numero 43 di questa rivista (giugno '87) è stato pubblicato un primo "approccio" all'uso di più di una pagina grafica nel C/128.

Si suggeriva, in quell'articolo, il ricorso ad un buffer (memoria temporanea) in cui immagazzinare provvisoriamente le nostre schermate oltre ad alcune routine l.m. per i trasferimenti di queste dal buffer alla loro sede naturale e viceversa.

Il metodo descritto può senz'altro trovare una sua valida applicazione in determinate circostanze, ma se in un nostro programma volessimo effettuare immediati cambiamenti tra una schermata e l'altra, magari per simulare un animazione, ci troveremmo di fronte agli inevitabili inconvenienti legati alla fase di trasferimento: anzitutto il sia pur minimo ritardo legato all'esecuzione delle routine l.m.; inoltre la mancanza di "pulizia" tra una schermata e l'altra, dovuta ai vari passaggi tra il modo FAST e quello SLOW nonchè alla disabilitazione e riabilitazione dello schermo.

A questi inconvenienti va anche aggiunto lo spazio disponibile che, inevitabilmente, risulta limitato. A disposizione dei programmi Basic, infatti, rimarrebbe solo la zona compresa tra la fine dell'area riservata alla pagina grafica ($4001) e l'inizio dei buffer; ma anche la memoria RAM libera da interferenze Basic (da $1300 a $1BFF) sarebbe condizionata dalla presenza delle routine in linguaggio macchina.

L'alternativa

Come fare per ovviare a tutto questo? Semplice: basta eliminare i trasferimenti.

Con questo non vogliamo proporre di lasciar perdere l'uso di una seconda schermata, ma solo dire che il nostro C/128 dispone di un registro che, se opportunamente usato, con una sola POKE può consentire l'immediato "switch" tra una pagina grafica e l'altra, con i vantaggi che potete immaginare.

Il registro di cui parliamo è locato all'indirizzo $D506 (decimale 54534), e viene chiamato RCR, che sta per "Ram Configuration Register". Esso svolge svariate mansioni a seconda di come sono impostati i singoli bit che lo compongono. Ciò che interessa in questa sede riguarda la possibilità di modificare il tipo di RAM "visibile" dal processore grafico a 40 colonne, il famoso VIC.

Come già saprete, nelle varie conformazioni di memoria possibili del C/128 (banchi) sono riscontrabili solo due tipi di RAM, dette RAM 0 e RAM 1, ma è la prima che normalmente viene usata per le operazioni di lettura e scrittura della schermata grafica.

Se, però, si modifica il valore presente in $D506 (54534) in modo da porre a 1 il bit numero 6, il VIC andrà a leggere la grafica in RAM 1, non più in RAM 0. Questo vale sia per l'alta risoluzione che per lo schermo normale, ma solo per le operazioni di lettura.

In altre parole: se dopo aver modificato il valore di RCR scrivete o disegnate qualcosa, l'immagine verrà sempre depositata nelle locazioni grafiche di RAM 0, ma visibile sarà l'area di RAM 1.

Avrete già capito che, in pratica, disponiamo di un vero e proprio interruttore grafico.

Avendo due schermate, una allocata normalmente in RAM 0 (BANK 0 o anche BANK 15) e una in RAM 1 (BANK 1), basterà impartire...

Poke 54534, Peek (54534) Or 64

...per vedere la seconda, e...

Poke 54534, Peek (54534) And 191

...per tornare alla prima. Se non dovete manipolare il registro RCR per altri motivi, Poke

Una porzione di schermo in hi-res, pur se di piccole dimensioni, rappresenta, in realtà decine di byte

54534,68 mostrerà la schermata in BANK 1 mentre Poke 54534,4 riporterà il registro al suo valore di default.

Come fare

Per non restare nel campo teorico, analizziamo con degli esempi i passi necessari per attuare quanto finora detto.

Innanzitutto, creata una prima schermata, è necessario trasferirla "una tantum" in RAM 1. Per far ciò è sufficiente una banale routine in linguaggio macchina, come quella che installa il caricatore Basic del listato 1, e della quale potete vedere, a parte, il disassemblato. La routine, grazie alla sua brevità, è allocata a partire da $27D (637 dec.) per lasciare libera l'area, da $1300 in poi, per eventuali altri usi.

Scambiare due aree di memoria tra di loro è un'impresa veloce e semplice se si opera in l.m.

La stessa routine, tuttavia, può essere trasferita in quelle locazioni, dato che, una volta usata, non serve più (a meno che non si voglia allocare una nuova pagina grafica) e può lasciare il posto ad altre routines.

Prima di essere utilizzata è necessario proteggere dall'interferenza delle variabili del Basic l'area RAM di BANK 1 che sarà usata per il trasferimento; questo compito viene svolto dalla riga 140 del listato caricatore.

Per i vostri usi, controllate sempre che le Poke che modificano i puntatori di inizio dell'area variabili si trovino tutte nella stessa riga, e che questa venga eseguita sempre prima di qualunque altra istruzione Basic.

La routine che vedete disassemblata trasferisce una schermata in alta risoluzione; se invece dovete utilizzarla per lo schermo normale, basterà entrare in monitor e modificare direttamente l'istruzione 0281 in LDA #$04, e la 029C in CMP #$08.

Gli esempi

Con il listato 2 possiamo vedere in azione il nostro bravo "interruttore" purchè, ovviamente, prima del RUN si sia caricata in memoria la routine l.m. appena descritta.

Il breve programma crea innanzitutto una schermata in hi-res con un quadrato vuoto ed un cerchio pieno; successivamente (riga 180) manda in esecuzione la routine l.m. che provvede al trasferimento in BANK 1 del disegno appena effettuato.

Cancellato lo schermo hi-res (riga 190), si ridisegnano le stesse figure nelle stesse posizioni, ma stavolta sarà il quadrato a essere pieno ed il cerchio vuoto.

A questo punto, premendo i tasti "1" oppure "2" le righe 250-260 modificheranno il contenuto di $D506 (dec. 54534), mostrando lo schermo desiderato.

Se provate a premere i tasti suddetti in rapidissima sequenza, vedrete come lo switch non pone alcun problema di velocità, rendendo l'illusione di operare con l'Amiga (si fa per dire!).

Premendo lo spazio si torna allo schermo normale, ed è basilare (riga 310) rimettere a posto il registro RCR, o si perderebbe il controllo dello schermo se in quel momento fosse attiva la RAM 1. Lo stesso avverrebbe se un errore (o la pressione del tasto stop) bloccasse il programma in questa configurazione, per cui è consigliabile usare sempre un'istruzione TRAP come quella presente in riga 140.

Anche in bassa risoluzione

Lo stesso algoritmo è seguito per il programma del listato-demo 3, il quale, però, non accede alla hi-res.

Si tratta di un semplice esempio sulla possibilità di creare un'animazione, in questo caso ponendo in alto a sinistra una diagonale di sferette [CHR$(209) di riga 170) che in uno schermo andrà verso destra e nell'altro verso sinistra. Alternando le due videate con adeguati cicli di ritardo (righe 240-250) si può avere l'illusione del movimento.

Prima di mandare in esecuzione il programma va sempre caricata in memoria la routine di trasferimento LM, in cui però siano apportate le modifiche descritte agli indirizzi di inizio e fine memoria schermo, in quanto tutto si svolge nella "solita" area da $0400 - $0800 di immagazzinamento del video in bassa risoluzione.

Su queste basi potete sbizzarrirvi a creare gli effetti che più vi aggradano, badando sempre di tenere più "alto" l'inizio dell'area di BANK 1 riservata alle variabili del Basic.

Ma questo, ormai, dovreste già saperlo.

```
100 REM ATTIVAZIONE=BANK15:SYS637
110 REM POKE 54534,4  = SCHERMO 1
120 REM POKE 54534,68 = SCHERMO 2
130 :
140 POKE 47,1: POKE 48,64:POKE 49,1:
    POKE 50,64:POKE 51,1:POKE 52,64
160 FORX=1TO36:READA:POKE636+X,A:NEXT
170 :
180 DATA 162,000,134,250,169
190 :     DATA 028: REM HI-RES
200 REM  DATA 004: REM LOW-RES
210 DATA 133
220 DATA 251,160,000,177,250,162,127
230 DATA 142,000,255,145,250,162,000
240 DATA 142,000,255,200,208,239,230
250 DATA 251,165,251,201,064,208,229
260 DATA 096
270 REM DIGITARE LA RIGA 190 OPPURE
280 REM LA 200 A SECONDA DEI CASI
```

```
100 REM DEMO SPLIT HI-RES PER C/128
110 :
120 REM CARICARE PRIMA ROUTINE LM
130 :
140 TRAP 290
150 GRAPHIC1,1:BOX1,20,60,100,140,0
160 COLOR1,5
170 CIRCLE1,160,100,40,40:PAINT1,165,105
180 BANK15:SYS637
190 SCNCLR1:BOX1,20,60,100,140,0
200 PAINT1,25,70
210 COLOR1,16:CIRCLE1,160,100,40,40
220 GRAPHIC0,1:PRINT"<1>-<2> SCHERMI"
230 PRINT"<SPAZIO> ESCE"
240 GETKEYA$
250 IFA$="1"THENGRAPHIC1:POKE54534,68
260 IFA$="2"THENGRAPHIC1:POKE54534,4
270 IFA$=CHR$(32)THEN310
280 GOTO240
290 BANK15:POKE54534,4:GRAPHIC0,1
300 PRINTERR$(ER):HELP:END
310 GRAPHIC0:POKE54534,4:END
```

```
100 REM DEMO BASSA RISOLUZIONE
110 :
120 REM MODIFICARE DAPPRIMA GLI
130 REM INDIRIZZI DELLA ROUTINE LM
140 :
150 TRAP 270
160 SCNCLR:FORX=1TO7
170 PRINTCHR$(209)+CHR$(17);:NEXT
180 BANK15:SYS637
190 SCNCLR:PRINTSPC(6);
200 FORX=1TO7
210 PRINTCHR$(209);CHR$(17);
220 PRINTCHR$(157);CHR$(157);:NEXTX
230 FORX=1TO50:POKE54534,68
240 FORY=1TO60:NEXTY
250 POKE54534,4:FORY=1TO60:NEXTY
260 NEXTX:END
270 POKE54534,4:PRINTERR$(ER):HELP
280 END
```

DISASSEMBLATO GRAPHIC SWITCH

```
027D  LDX #$00     Prepara due locazioni
027F  STX $FA      di pagina zero con
0281  LDA #$1C     l'indirizzo di inizio
                   (se hi-res)
0281  LDA #$04     l'indirizzo di inizio
                   (se low-res)
0283  STA $FB      trasferimento ($1C00)
-----------------------------------------
0285  LDY #$00
0287  LDA ($FA),Y Legge locazione,
0289  LDX #$7F     passa in BANK 1
028B  STX $FF00    e riscrive nella
028E  STA ($FA),Y stessa posizione.
0290  LDX #$00
0292  STX $FF00    Setta BANK 15.
-----------------------------------------
0295  INY
0296  BNE $0287    Ripete fino
0298  INC $FB      all'indirizzo
029A  LDA $FB      esadecimale $4000.
029C  CMP #$40
029E  BNE $0285
-----------------------------------------
02A0  RTS          Return.
```

DIVENTARE CINEASTI CON IL COMMODORE 64

Un'eccezionale procedura per C/64 che permetterà di creare facilmente i vostri cartoni animati

di **Valentino Spataro**

C'era una volta Walt Disney, tanti fogli di carta, matite e forbici. Poi arrivarono i giapponesi che, avvalendosi dei computer, erano in grado di produrre numerosi cartoni animati in poco tempo. E' inoltre recente la notizia secondo cui l'Amiga è utilizzato per visualizzare, sul monitor, quello che poi accadrà sul set cinematografico.

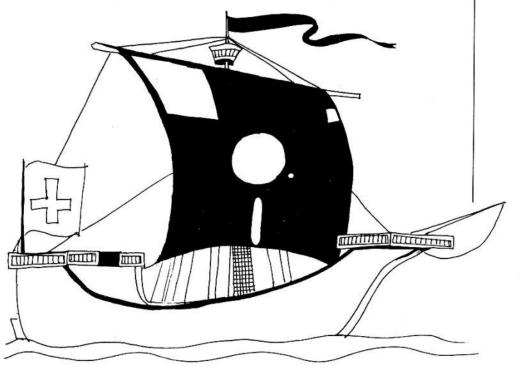
Per far fronte alla sempre maggiore richiesta di utilizzare il calcolatore nel campo della grafica è nato MOVIE MAKER, per scoprire tutto quello che non sapevate sull'animazione, e che non avete mai osato chiedere.

A che serve il programma

E' bene precisare subito che il programma richiede l'uso del joystick in porta 2 e le routinè grafiche di Toma; può essere utile a chi abbia la necessità di memorizzare numerose immagini, indipendentemente dalla esigenza di vederle in rapida successione.

Può anche essere utilizzato per aggiungere, nei giochi, brevi sequenze animate.

Prima di mettersi all'opera, è bene ricordare tutte le operazioni necessarie da svolgere in successione:

Sono necessarie le routine di Toma per operare con il programma proposto

• Caricare e lanciare le routine grafiche di Toma.
• Digitare, salvare e verificare il listato Basic.
• Dare il Run ed attendere il tempo necessario affinchè sia effettuato il caricamento dei Data relativi alle routine "Input Dati", "Mem figura", "Richiama figura", "Transfer mem".
• Nel caso vengano segnalati errori di digitazione, correggere e salvare nuovamente il listato.

Il programma

Il programma è semplicissimo da usare perchè dotato di un ricco menu di scelta

Lo scopo, come abbiamo detto, è quello di mostrare in successione progressiva (o regressiva), e con tutte le pause desiderate, una sequenza di immagini che, da questo momento, chiameremo, per analogia con il mondo del cinema, "fotogrammi", definiti precedentemente dall'utente.

Dato il RUN comparirà sullo schermo Hi-Res un rettangolo lampeggiante; muovendo il joystick potrete variarne le dimensioni. Premendo il pulsante Fire "fisserete" le dimensioni dei fotogrammi del vostro film.

Siccome le immagini sono composte da pixel, ed i pixel occupano memoria, maggiore è il numero di pixel contenuti nel fotogramma, minore sarà il numero di fotogrammi memorizzabili.

Premendo F1 durante la determinazione delle misure del fotogramma, verrà visualizzato il numero, approssimativo, di fotogrammi memorizzabili. La grandezza massima del fotogramma è comunque controllata dallo stesso programma.

Premuto il pulsante Fire vedrete alcuni punti lampeggianti che delimitano gli estremi di un rettangolo che potrete spostare usando il joystick; rappresenta la porzione di video in cui vedrete in seguito, durante l'uso del programma, la sequenza dei fotogrammi.

Vi consigliamo, almeno all'inizio, di inserire questa "finestra" in una porzione di schermo non ricoperta dal rettangolo precedente (che continuerà sempre ad esser visibile) ed il più possibile in basso a sinistra.

Se, a causa delle dimensioni del fotogramma, è inevitabile una sovrapposizione, mettete il riquadro nella zona che più vi aggrada, senza preoccuparvi di altro perchè il programma contempla questa possibilità.

Nonostante tutto è sempre preferibile non sovrapporre le due zone; le prime volte che userete il programma limitatevi a confermare le posizioni di default del programma.

Premuto ancora una volta Fire vedrete lampeggiare la scritta EDIT: a questo punto, servendovi della manopola del joystick, potrete selezionare l'opzione desiderata, premendo Fire confermarla.

Descriviamo ora tutte le opzioni disponibili:

EDIT

Scegliendo questa opzione si passa al sub-menu, posto in basso, tra cui scegliere DRAW per disegnare, DELETE per cancellare, INVERT per invertire ciò che è posizionato al di sotto del cursore, DROP per tratteggiare, MOVE per muovere il cursore senza disegnare, RUN per muoverlo velocemente, CLEAR per pulire la finestra dove si disegna, EXIT per tornare al menu principale.

MEMO

Permette di memorizzare il fotogramma disegnato nel riquadro, per utilizzarlo in seguito.

MOVIE

Permette di vedere i fotogrammi disegnati e memorizzati adeguatamente. Viene visualizzato il fotogramma e, contemporaneamente, il relativo numero ordinale lampeggiante. Premendo il joystick verso destra il filmato viene visto in avanti, verso sinistra all'indietro. Non muovendo il joystick si ferma l'immagine. Si ottiene così l'effetto moviola.

SAVE

Permette di salvare un fotogramma su disco. E' possibile scegliere il fotogramma desiderato muovendo il joystick come per MOVIE e premere Fire per confermare la scelta; viene poi richiesto il nome del file: se non viene immesso nessun nome il programma torna al menu, altrimenti procede alla registrazione. E' possibile registrare un solo fotogramma per volta; ciò permette, in seguito, di richiamare in memoria fotogrammi anche appartenenti a diversi filmati.

LOAD

Viene richiesto il nome del file da cui richiamare i dati del fotogramma. Se il foto-

gramma richiesto è di dimensioni diverse da quelle definite, il programma non carica il file e chiede l'operazione di reset. Durante il reset viene già definito il formato richiesto del fotogramma; basta quindi confermarlo premendo Fire senza variarne le dimensioni. Se non è necessario il reset, i dati richiamati vengono aggiunti a quelli già presenti con la verifica della disponibilità di memoria; in caso contrario il file non viene caricato.

COPY

Questa opzione serve per modificare un fotogramma. In questo caso il fotogramma scelto viene ricopiato nell'angolo in alto a sinistra per essere poi modificato. Naturalmente il disegno precedentemente presente in questo angolo viene cancellato.

L'opzione può essere comoda anche nel caso in cui il fotogramma sia così grande che non possa coesistere assieme alla finestra in cui si vedono i fotogrammi senza sovrapporsi. Avvalendoci di questa opzione potremo infatti limitarci a modificare piccole parti di un fotogramma, senza esser costretti a disegnarlo nuovamente per intero (utile nel caso di cartoni animati).

OTHER

Opzione per future espansioni, che può tornare utile per ulteriori ampliamenti.

RESET

Può essere utilizzata nel caso di un load (vedi sopra) o semplicemente per ridefinire le dimensioni del fotogramma. Cancella tutti i fotogrammi contenuti in memoria.

Da ricordare che non è possibile modificare il contenuto di un fotogramma in memoria: è però possibile richiamarlo (con COPY), modificarlo (con EDIT) e memorizzarlo (con MEMO) come un fotogramma di numero diverso. Se poi, in fase di visualizzazione, non interessa qualche fotogramma, basta registrare tutti gli altri, resettare e richiamarli.

La stessa procedura torna utile nel caso si voglia invertire la sequenza di qualche fotogramma. E' da ricordare, inoltre, che il programma è nato per essere facilmente personalizzato. Chi vuole limitarsi all'uso del programma può tranquillamente terminare la lettura di queste note, mentre chi vuole conoscere la tecnica di programmazione usata e scoprire i segreti del programma, per modificarlo a piacere, non si perda d'animo ma, pacchetto di popcorn alla mano, si rilassi e proceda nella lettura.

"Dentro" il programma

La configurazione della memoria durante il funzionamento del programma è la seguente:

...in cui LU è la lunghezza e LOC la locazione a partire dalla quale i dati vengono memorizzati

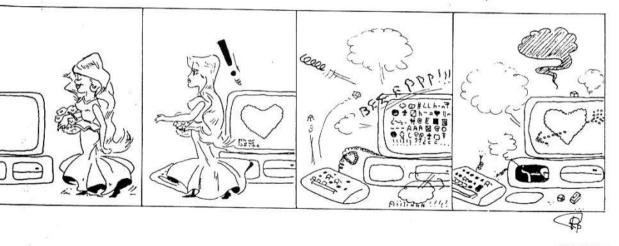
5) le locazioni \$02-\$0A e \$F3-\$F4 sono usate dalle routine in l.m.

6) le routine, per disabilitare e abilitare le ROM del Basic e del Kernal, sfruttano subroutine già presenti nelle routine di Toma (vedi disassemblato pubblicato sul fascicolo "Commodore speciale").

Le routine in l.m.

Descriviamo ora le quattro routine in l.m.:

- la prima serve a determinare l'inizio della

Anche chi è digiuno di Assembly potrà utilizzare il programma pubblicato

porzione di schermo desiderata.
• la seconda prende i dati della pagina grafica e li posiziona a partire da una data locazione di memoria.
• la terza prende i dati a partire da una locazione e li mette nella pagina grafica, nella porzione di schermo indicata.
• la quarta trasferisce i dati da un'area di memoria ad un'altra.

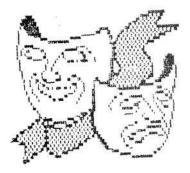
La sintassi corrispondente alle routine è la seguente:

SYS A(0),X,Y

dove X e Y rappresentano le coordinate dell'angolo nord-ovest della finestra in cui compariranno i fotogrammi; X+LX deve essere maggiore di zero e minore di 39, Y+LY deve esser compreso tra 1 e 23. Per semplicità, infatti, si è preferito definire l'area gestibile con "modulo" di 8 pixel, corrispondente all'area richiesta da un carattere.

Una Sys è "pericolosa" per natura: digitate i Data con la massima attenzione

- *2048-17000 programma in Basic*
- *17001-25192 8Kbyte a disposizione delle variabili*
- *25193-25692 l.m. routine*
- *25693-49152 memoria Ram per i dati dei fotogrammi*
- *49153-53248 routine di Toma*
- *53248-61440 bitmap*

Bisogna ancora precisare alcune cose:
1) affinchè le variabili del Basic non si sovrappongano al programma in l.m. e ai dati dei fotogrammi, il limite della memoria RAM è stato abbassato a 8 Kbyte dopo la fine del programma (puntatori 55/56).

2) modificando il programma le routine in l.m. possono venir spostate: questo non è un problema perchè le routine sono rilocabili(!) e gli indirizzi di partenza vengono ricalcolati e inseriti nella matrice A(n) dallo stesso programma.
3) Come tutti sapranno la memoria RAM utilizzabile dall'utente in ambiente Basic arriva alla locazione 40960 (da cui, sottraendo 2048, cioè il numero di byte della pagina zero più la memoria video, si ottiene 38912, numero di byte liberi all'accensione).
Da linguaggio macchina, però, è possibile sfruttare anche gli 8K di RAM successivi pur essendo "nascosti" dalla ROM destinata all'interprete Basic. In questo modo sono a disposizione 8K di memoria in più per dati di ogni genere.
Questo fatto genera difficoltà solo in fase di registrazione su supporto magnetico: dovendo disabilitare la ROM in questione per prelevare i dati memorizzati, non è più possibile servirsi del Basic, appunto perchè la ROM disabilitata è proprio quella dell'interprete Basic.
Bisogna quindi trasferire i dati da questa zona di memoria, tramite routine l.m., in un'area accessibile al Basic. A questo punto l'area di memoria più grande rimasta libera è quella del video in bassa risoluzione; qui vengono infatti trasferiti i dati in gruppi di 1000 byte (o meno) a seconda della lunghezza dei dati del fotogramma, e di volta in volta "spediti" al drive.
4) la lunghezza dei dati di un fotogramma si calcola con le linee di programma...

```
10 lu = peek(5) + peek(6)*256 - loc
20 if peek(243) = 51 then lu = lu+2
30 print lu
```

*SYS A(1),LX*8,LY,LOC*
dove LX e LY sono le dimensioni del fotogramma e LOC rappresenta la locazione a partire dalla quale verranno memorizzati i dati contenuti nel rettangolo in alto a sinistra di dimensioni LX e LY.

*SYS A(2),LX*8,LY,LOC*
dove LX e LY sono le variabili di cui abbiamo già parlato e LOC è la locazione di partenza da cui prendere i dati per visualizzarli in pagina grafica. E' assolutamente indispensabile dare immediatamente prima la SYS A(0),X,Y per definire la posizione della finestra, pena l'inchiodamento del programma (LX compreso tra 1 e 30; LY tra 1 e 23).

SYS A(3)
richiede preventivamente l'impostazione da Basic della locazione di partenza da cui prendere i dati (in 02/03) la locazione di partenza dove verranno trascritti (04/05) ed il numero di dati da trasferire (06/07), sempre secondo la forma LB/LH (vedi disassemblato).

Come memorizza le immagini

Molti sapranno che i byte nella bit map sono disposti come nella tabella pubblicata.

RIGA N.	1	2	3	...	39
L	0	8	16	...	312
I	1	9	17		313
N	2	10	18		314
E	3	11	19		315
A	4	12	20		316
	5	13	21		317
1	6	14	22		318
	7	15	23		319
L	320	328	336	...	632
I	321	329	337		633
N	322	330	338		634
E	323	331	339		635
A	324	332	340		636
	325	333	341		637
2	326	334	342		638
	327	335	343		639
...					

Caratteristica peculiare della routine n.2 è il fatto di non memorizzare tutti i byte contenuti nell'intera area bit-map del fotogramma.

La routine carica infatti il primo byte, controlla se è uguale a 0 (cioè se in quella porzione di schermo non compare alcun pixel "acceso") nel qual caso conta il numero di byte eventualmente uguali a zero che compaiono dopo il primo e mette il numero di byte uguali a zero nelle due locazioni successive; in seguito carica un altro byte e continua allo stesso modo.

Questo è stato realizzato per il semplice motivo che, generalmente, gli spazi "vuoti" di una figura sono più numerosi degli spazi "pieni" e non vale la pena memorizzare lunghe sequenze di zeri.

Ad esempio, un fotogramma può esser rappresentato da una sequenza di dati come la seguente...

126, 33, 0, 34, 1, 34, 1

...che andrà interpretata, in fase di "decodifica", in questo modo:

nel byte 0 era presente il valore 126, nel byte 1 il 33; subito dopo (a partire, cioè, dal byte n.2), poichè compare uno zero, sarà presente un gruppo di byte nulli dato dal valore dei due byte successivi: 34+1*256=290. Proseguendo, poi, ne incontriamo uno uguale a 34 e un altro uguale a 1.

Si noti, quindi, che i due 34 e i due 1 non hanno lo stesso significato: nel primo caso, essendo posti subito dopo uno zero, indicano quanti zeri sono presenti "dopo"; nel secondo caso, invece, rappresentano il contenuto di un byte.

La tecnica di programmazione illustrata, pertanto, consente di memorizzare un'immagine-fotogramma (prevalentemente "vuoto") in soli sette byte, là dove, volendola memorizzare per intero, avrebbe richiesto ben 294 byte.

Tale modo di operare, e la disposizione dei byte in memoria, possono tornare utili per chiunque voglia aggiungere una routine per la stampa dei singoli fotogrammi, routine che non è stata realizzata per ragioni di brevità.

Il programma provvede a compattare i [illegible] fotogrammi

Come "leggere" il listato

Il programma in Basic si divide, sostanzialmente, in due parti: nella prima vengono inizializzate le variabili, caricate le routine in l.m.; nella seconda, invece, sono presenti varie subroutine molto interessanti e linee DATA che possono sembrare lunghe, ma in realtà sono poche considerando le prestazioni che permettono.

Si consiglia di dare una sguardo alle subroutine, soprattutto per quanto riguarda quella che individua i movimenti del joystick, quella che visualizza il menu e quella per la scelta delle opzioni.

Per facilitare la comprensione del programma segue la lista delle variabili utilizzate.

Il programma, ampiamente commentato, risulterà certamente meno difficile dopo la ri-lettura del presente articolo.

LX,LY: dimensioni dei fotogrammi

X1,Y1: coordinate dell'angolo in alto a sinistra della finestra in cui si succedono i fotogrammi

NL: numero dei fotogrammi memorizzati

SP: step per lo spostamento del cursore nella finestra di edit

W: n. opzione scelta

LD: locazione inizio RAM a disposizione dei dati dei fotogrammi

Q: n. fotogramma scelto

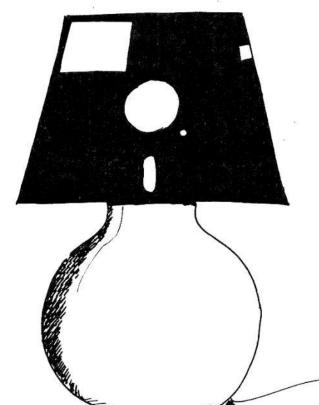
A(): contiene gli indirizzi di partenza delle routine in l.m. (utile per non esser costretti a ricordare ogni volta l'indirizzo di partenza)

A$(N): contiene i messaggi relativi alle N opzioni

B$(): contiene i messaggi che di volta in volta compaiono

NL(X): contiene la lunghezza di ogni fotogramma. E' fissato uguale a 100, ma può essere incrementato, a patto di aumentare, se necessario, lo spazio di 8K destinato alle variabili.

Oltre il programma

Il package si presta a sofisticazioni notevoli, tra cui suggeriamo:
• stampa dei fotogrammi.
• registrazione su disco di tutti i fotogrammi con un unico comando.
• caricamento con un solo comando di tutti i fotogrammi appartenenti a uno stesso filmato.
• determinazione dei vari ordini in cui vedere i fotogrammi.
• miglioramento delle funzioni di edit (per esempio possibilità di scrivere e di aggiungere il tracciamento di linee e di figure geometriche).
• possibilità di operare un "merge" tra i vari fotogrammi: basta infatti modificare la routine n. 3 come suggerito perchè il fotogramma successivo non cancelli quello precedente ma effettui una sovrapposizione (operazione di OR logico). Attenti, però, a cambiare il valore di controllo dei dati del gruppo n.3.

SCHEDA TECNICA

Software applicativo per grafica e giochi

Hardware richiesto: C/64, Disk drive 1541 (o compatibile) joystick oppure mouse.

Programma non adattabile ad altri computer Commodore

Richiede routine grafiche di Toma

Consigliato agli esperti

Anche il programma pubblicato in queste pagine (oltre ad alcuni file applicativi e ad una versione semplificata delle routine grafiche di Toma) è contenuto nel disco "Directory" di questo mese.

ROUTINE N.1:
Determinazione della locazione della bit-map a partire dalla quale trascrivere i dati

```
9471 jsr $b7f1   ;prende x
9474 stx $04     ;e lo mette in 04
9476 jsr $b7f1   ;prende y
9479 stx $05     ;e lo mette in 05
947b lda #$00    ;azzera 'a' e 'x'
947d tax         ;prepare 02 e 03 come
947e sta $02     ;puntatori alla bitmap
9480 lda #$e0    ;
9482 sta $03     ;
9484 lda $05     ;carica il n. di linee
9486 cmp #$00    ;e' uguale a 0 ?
9488 beq $949d   ;se si salta
948a clc         ;no modifica puntatori
948b inc $03     ;alla linea successiva
948d lda #$40    ;sommando 320
948f adc $02     ;
9491 sta $02     ;
9493 bcc $9497   ;
9495 inc $03     ;
9497 dec $05     ;decrementa il n. linee
9499 cpx $05     ;e' uguale a 0 ?
949b bne $948a   ;no continua
949d lda $04     ;carica il n. di righe
949f cmp #$00    ;e' uguale a 0 ?
94a1 beq $94b6   ;se si' torna a basic
94a3 lda #$08    ;se no somma 8
94a5 clc         ;ai puntatori
94a6 adc $02     ;
94a8 sta $02     ;
94aa bcc $94ae   ;
94ac inc $03     ;
94ae dec $04     ;decrementa n. righe
94b0 lda $04     ;carica n.righe
94b2 cmp #$00    ;lo confronta con 0
94b4 bne $94a3   ;se non e' =0 continua
94b6 rts         ;altrimenti esce
```

ROUTINE N.2:
Per trasferire i dati della bitmap in ram

```
94b7 jsr $b7f1   ;prende lx*8
94ba stx $07     ;e lo mette in 07
94bc jsr $b7f1   ;prende ly
94bf stx $08     ;e lo mette in 08
94c1 jsr $aefd   ;prende loc di partenza
94c4 jsr $ad8a   ;da cui iniziare a
94c7 jsr $b7f7   ;riporre i dati presi
94ca lda $14     ;bitmap
94cc sta $05     ;crea 05 e 06 come
94ce lda $15     ;puntatori
94d0 sta $06     ;
94d2 jsr $c55e   ;disab. kernal (v.toma)
94d5 jsr $c958   ;disab. basic (v.toma)
94d8 lda $08     ;prende il n. di linee
94da sta $f4     ;e lo mette in f4
94dc lda #$00    ;azzera 'a'
94de tax         ;e 'x'
94df sta $03     ;e lb puntat.a bitmap
94e1 sta $f3     ;e f3
94e3 lda #$e0    ;imposta hb del punta-
94e5 sta $04     ;tore alla bitmap
94e7 ldy #$00    ;azzera 'y'
94e9 lda ($03),y;prende byte da bitmap
94eb cmp #$00    ;e' uguale a zero ?
94ed beq $9538   ;se si' salta
94ef lda $f3     ;se f$ impostato allora
94f1 cmp #$33    ;e' finita una serie di
94f3 beq $9526   ;0 e modifica puntatori
94f5 ldx #$00    ;azzera 'x'
94f7 lda ($03),y;riprende byte dabitmap
94f9 sta ($05),x;e lo mette in ram
94fb inc $05     ;aumenta puntatore
94fd lda $05     ;della ram
94ff cmp #$00    ;
9501 bne $9505   ;
9503 inc $06     ;e aumenta 'y' per pun-
9505 iny         ;tare a byte successivo
9506 cpy $07     ;in bitmap e controlla
9508 bne $94e9   ;il n. di righe
950a inc $04     ;se riga finita aggiun-
950c clc         ;ge 320 a puntatore
950d lda #$40    ;della bitmap
950f adc $03     ;per andare alla linea
9511 sta $03     ;successiva
9513 bcc $9517   ;
9515 inc $04     ;
9517 dec $f4     ;decrementa il n. di
9519 lda $f4     ;linee e controlla se
951b cmp #$00    ;sono finite
951d bne $94e7   ;se no continua
951f jsr $c56d   ;se si riabilita basic
9522 jsr $c960   ;e kernal
9525 rts         ;e esce
```

ripristino puntatori a ram dati dopo una serie di zeri

```
9526 lda #$00    ;azzera f3
9528 sta $f3     ;e incrementa puntatore
952a inc $05     ;a ram dati
952c lda $05     ;
952e cmp #$00    ;
9530 bne $9534   ;
9532 inc $06     ;
9534 dey         ;e salta a ulteriore
9535 clc         ;incremento puntatori
9536 bcc $94fb   ;

9538 lda $f3     ;prende f$
953a cmp #$33    ;e'uno 0 di una serie?
953c beq $9555   ;se si incrementa cont.
```

imposta contatore numero di zeri

```
953e sty $02     ;salva il n. di righe
9540 ldy #$00    ;azzera 'y'
9542 tya         ;e 'a' e i 3 bytes segg
9543 sta ($05),y;avvisa inizia serie 0.
9545 iny         ;
9546 sta ($05),y;lb n. di zeri
9548 iny         ;
9549 sta ($05),y;hb n. di zeri
954b lda #$33    ;e imposta f3 per av-
954d sta $f3     ;vertire inizio serie 0
954f ldy $02     ;carica il n. di righe
9551 dey         ;
9552 clc         ;e ricominicia
9553 bcc $94fb   ;
```

routine di incremento del contatore del numero dei zeri

```
9555 sty $02     ;salva il n. di righe
9557 ldy #$00    ;azzera 'y'
9559 lda ($05),y;carica lb contatore
955b tax         ;
955c inx         ;e lo incrementa di 1
955d txa         ;
```

```
955e sta ($05),y;e lo memorizza
9560 cmp #$00    ;
9562 bne $956c   ;
9564 iny         ;idem per hb
9565 lda ($05),y;idem
9567 tax         ;
9568 inx         ;idem
9569 txa         ;
956a sta ($05),y;idem
956c ldy $02     ;
956e clc         ;
956f bcc $9505   ;e continua rout. prin
```

ROUTINE N.3:
Prende i dati memorizzati a partire da una locazione e li mette nella bitmap

```
9571 jsr $b7f1   ;prende lx*8
9574 stx $07     ;e lo mette in 07
9576 jsr $b7f1   ;prende ly
9579 stx $08     ;e lo mette in 08
957b jsr $aefd   ;prende loc di partenza
957e jsr $ad8a   ;della ram dati
9581 jsr $b7f7   ;
9584 lda $14     ;e li mette in 04 e 05
9586 sta $04     ;
9588 lda $15     ;
958a sta $05     ;
958c jsr $c55e   ;disab kernal v. toma
958f jsr $c958   ;disab basic v. toma
9592 lda $08     ;mette in f4 il numerc
9594 sta $f4     ;di righe
9596 lda #$00    ;azzera f3
9598 sta $f3     ;
959a ldy #$00    ;e 'y'
959c lda $f3     ;e' in contemporanea
959e cmp#$33     ;una serie di zeri ?
95a0 beq $95d9   ;se si' salta
95a2 ldx #$00    ;se no azzera 'x'
95a4 lda ($04),x;prende un dato da ram
95a6 cmp #$00    ;e' un byte = 0 ?
95a8 beq $9608   ;se si' inizia serie 0
95aa nop         ;sostituibili con
95ab nop         ;ora ($02),y per merge
95ac sta ($02),y;e mette in bitmap
95ae inc $04     ;incrementa puntatore
95b0 lda $04     ;alla ram dati
95b2 cmp #$00    ;lb e eventualmente hb
95b4 bne $95b8   ;
95b6 inc $05     ;
95b8 iny         ;incrementa n. di righe
95b9 cpy $07     ;e' finito /
95bb bne $959c   ;se no continua
95bd inc $03     ;se si somma 320 a pun-
95bf clc         ;tatore bitmap
95c0 lda #$40    ;per passare alla
95c2 adc $02     ;linea successiva
95c4 sta $02     ;
95c6 bcc $95ca   ;
95c8 inc $03     ;
95ca dec $f4     ;e decrementa n. linee
95cc lda $f4     ;
95ce cmp #$00    ;sono finite ?
95d0 bne $959a   ;se no continua
95d2 jsr $c56d   ;se si riabilita
95d5 jsr $c960   ;kernal e basic
95d8 rts         ;e esce
```

decremento contatore numero di bytes=0 da mettere in bitmap

```
95d9 dec $09     ;decrementa n. bytes =0
95db lda $09     ;rimanenti
95dd cmp #$00    ;
95df bne $95e3   ;se necessario anche hb
95e1 dec $0a     ;
95e3 lda $09     ;e' finito il n. di 0 ?
95e5 cmp #$ff    ;(parte lb)
95e7 bne $9601   ;se no continua
95e9 lda $0a     ;e' finito il n. di 0 ?
95eb cmp #$ff    ;(parte hb)
95ed bne $9601   ;se no continua
95ef lda #$00    ;se si azzera f3 (fine
95f1 sta $f3     ;serie di 0) e aumenta
95f3 inc $04     ;puntatore a ram dati
95f5 lda $04     ;
95f7 cmp #$00    ;
95f9 bne $95fd   ;
95fb inc $05     ;e salta a altro incre-
95fd dey         ;mento di puntatori
95fe clc         ;
95ff bcc $35ae   ;
9601 lda #$00    ;mette un byte =0
9603 sta ($02),y;(sostituib. con 2 nop)
9605 clc         ;in bit map e continua
9606 bcc $95b8   ;
```

procedura eseguita all'inizio di una serie di bytes = 0

```
9608 lda #$33    ;imposta f3 per inizio
960a sta $f3     ;serie di 0
960c inc $04     ;e incrementa puntatori
960e lda $04     ;a ram dati
9610 cmp #$00    ;lb e hb
9612 bne $9616   ;
9614 inc $05     ;
9616 sty $06     ;salva numero righe
9618 ldy #$00    ;azzera 'y'
961a lda ($04),y;mette in 09 lb n. 0
961c sta $09     ;
961e iny         ;
961f lda ($04),y;mette in 0a hb n. 0
9621 sta $0a     ;
9623 ldy $06     ;
9625 clc         ;e salta
9626 bcc $95d9   ;
```

ROUTINE N.4:
Trasferimento dati
dall'area puntata da 02/03
verso l'area puntata da 04/05
per un n. di bytes indicato da 06/07

```
9628 jsr $c55e   ;disab. basic
962b ldy #$00    ;azzera 'y'
962d ldx #$00    ;e 'x'
962f lda ($02),y;prende da
9631 sta ($04),y;mette in
9633 iny         ;incrementa e
9634 cpy $06     ;confronta per vedere
9636 bne $9640   ;se ha finito
9638 cpx $07     ;
963a bne $9640   ;
963c jsr $c56d   ;se si abilita basic
963f rts         ;e esce
9640 cpy #$00    ;deve incrementare hb
9642 bne $962f   ;del n. dati da trasf.?
9644 inx         ;si
9645 inc $03     ;
9647 inc $05     ;
9649 clc         ;
964a bcc $962f   ;e riprende
```

```
100 REM ** MOVIE MAKER **
110 REM ** BY SPATARO  **
120 REM INIZIALIZE
130 ←TEXT0,1:PRINTCHR$(147)"CARICA LM ROUTINES"
140 ←CLEAR:←COLOR1:POKE55,PEEK(45):POKE56,PEEK(46)+32:CLR:REM 8K PER VAR
150 LOC=PEEK(55)+(PEEK(56)+1)*256:LD=LOC+500
160 W=0:N=16:VI=53248:DIMA$(N),B$(1),NL(100)
170 GOSUB1630:POKE53280,3:CR$=CHR$(147)
180 ←GRAF0,1:GOSUB1170:POKEVI,30:POKEVI+1,50:SP=8:REM SPRITE
190 GOSUB1360
200 B$(0)="DEF REC":B$(1)="":GOSUB1380
210 X=-96:Y=35:IFLXTHENX=-160+LX*8:Y=100-LY*8
220 SP=8:VX=X:VY=Y:B=1:GOSUB1410
230 IFLX>30ORLY>24THEN:←CLEAR:GOTO190
240 TPMEMO=49152-LX*8*LY:S=1:W=0
250 VX=X+8:VY=Y
260 X=X+8:Y=Y:B=2:←INV1:GOSUB1600:←INV0:X1=(X+160)/8:Y1=(100-Y)/8-LY
270 IFX1<0ORY1<0ORX1+LX>40ORY1+LY>24THENVX=X:VY=Y:GOTO260
280 REM SCELTA MENU' PRINC.
290 GOSUB930:GOSUB1110:S=0:MAX=7:POKEVI+21,0:GOSUB950
300 W=W+1:ONWGOTO310,490,560,580,590,580,290,890
310 REM EDIT
320 POKEVI+21,255:S=S+1:GOSUB930:S=9:MAX=7:GOSUB950
330 ←INV0:SP=1:←COLOR1
340 IFW=1THEN:←COLOR0
350 IFW=2THEN:←INV1
360 IFW=3THENSP=2
370 IFW=5THENSP=8
380 IFW=6THENGOSUB910:GOTO320
390 IFW=7GOTO290
400 X=PEEK(VI):Y=147-PEEK(VI+1)
410 B=3:GOSUB1410:IFA<0THEN320
420 IFX<22THENX=22+LX*8
430 IFY>100THENY=100-LY*8
440 IFX>22+LX*8THENX=22
450 IFY<100-LY*8THENY=100
460 POKEVI,X:POKEVI+1,147-Y:IFW<4THEN:←PLOT X-182,Y,0
470 GOTO400
480 REM MEMO
490 LU=0:FORA=0TONL:LU=LU+NL(A):NEXT
500 IFLU>49152-LX*16*LYTHENB$(0)="":B$(1)="MEMORY FULL":GOSUB1260:GOTO290
510 SYSA(1),LX*8,LY,LD+LU:GOSUB930
520 NL=NL+1:NL(NL)=(PEEK(5)+PEEK(6)*256)-(LD+LU)+1
530 IFPEEK(243)=51THENNL(NL)=NL(NL)+1
540 GOTO290
550 REM MOVIE
560 GOSUB1300:GOTO290
570 REM ROUTINE COMUNE PER SAVE E LOAD E COPY
580 GOSUB1300
590 B$(0)="OK":IFW=6GOTO870:REM COPY
600 PRINTCHR$(147):←TEXT0,1:PRINTFL$:GOSUB930
610 FL$="":INPUT"NOME FILE";FL$:IFFL$=""GOTO850
620 IFW=5GOTO740:REM VA A LOAD
630 REM SAVE
640 OPEN1,8,15:OPEN2,8,2,FL$+",S,W":INPUT#1,A,A$
650 IFA<>0THENPRINTA$:CLOSE2:CLOSE1:GOTO610
660 PRINT#2,NL(Q):PRINT#2,LX:PRINT#2,LY
670 NN=INT(NL(Q)/1000):FORB=0TONN:PRINTCR$
680 A1=LD+LU+B*1000:A2=1024
690 LN=1000:IFNN=BTHENLN=NL(Q)-B*1000
700 A3=LN:GOSUB1210
710 FORA=0TOLN-1:A$=CHR$(PEEK(1024+A)):PRINT#2,A$;:NEXTA
```

```
720 NEXTB:GOTO840
730 REM LOAD
740 OPEN1,8,15:OPEN2,8,2,FL$+",S,R":INPUT#1,A,A$
750 IFA<>0THENPRINTA$:CLOSE2:CLOSE1:GOTO610
760 PRINTCHR$(147):INPUT#2,NL(NL+1):LU-0:FORA-0TONL:LU=LU+NL(A):NEXT:NL-NL+1
770 IFLU>49152-NL(NL)THENB$(0)="MEMORY FULL":NL=NL-1:GOSUB1260:GOTO290
780 INPUT#2,X,Y:IFX<>LXORY<>LYTHENZX=X:ZY-Y:B$(0)-"CHOOSE RESET TO LOAD":GOTO840
790 NN-INT(NL(NL)/1000):FORB=0TONN:PRINTCR$
800 A2=LD+LU+B*1000:LN=1000
810 IFNN-BTHENLN-NL(NL)-B*1000
820 FORA=0TOLN-1:GET#2,A$:Z=ASC(A$+CHR$(0)):POKE1024+A,Z:NEXTA
830 A1-1024:A3-LN:GOSUB1210:NEXTB
840 CLOSE2:CLOSE1
850 GOSUB1260:GOTO290
860 REM COPY
870 SYSA(0),0,0:SYSA(2),LX*8,LY,LD+LU:GOTO290
880 REM RESET
890 NL-0:LX-ZX:LY-ZY:←CLEAR:GOSUB1360:GOTO180
900 REM PULISCE FINESTRA DI EDIT
910 A$-"":FORA-1TOLX:A$-A$+" ":NEXT
920 FORA-0TOLY-1:←CHAR 0,A,0,A$:NEXT:RETURN
930 ←CHAR 34,W+S-1,0,A$(W+S-1):RETURN
940 REM ROUTINE SCELTA MENU'
950 W-0:FORA-1TO250:NEXT
960 B-3:GOSUB1410
970 ←CHAR 34,W+S,0,A$(W+S)
980 IFA-10THENW-W-1:IFW<0THENW-MAX
990 IFA-9THENW-W+1:IFW>MAXTHENW-0
1000 ←CHAR 34,W+S,1,A$(W+S)
1010 IFA=>0THEN960
1020 RETURN
1030 REM SCELTA FOTOGRAMMA
1040 B=3:GOSUB1410
1050 ←CHAR 35,18,0,"    ":REM4 SPAZI
1060 IFA=3THENQ=Q+1:IFQ>NLTHENQ=1
1070 IFA=7THENQ=Q-1:IFQ<0THENQ=NL
1080 ←CHAR 35,18,0,STR$(Q)+" "
1090 RETURN
1100 REM MEMORY STATUS
1110 LU=0:FORC=0TONL:LU=LU+NL(C):NEXT
1120 C-49152-LD-LU:MD=INT(C/(LX*LY*2))
1130 B$(0)="FREE : RAM"+STR$(C)+"  N.FIG."+STR$(MD)
1140 B$(1)="USED : RAM"+STR$(LU)+"  N. FIG."+STR$(NL)
1150 GOSUB1380:RETURN
1160 REM DATI DELLO SPRITE
1170 FORA-52096TO52160:POKEA,0:NEXT
1180 FORA-52096TO52096+15STEP3:POKEA,32:NEXT
1190 POKE52096+6,248:POKE53240,46:POKEVI+39,7:RETURN
1200 REM PROCEDURA PREPARAZIONE SYSA(3)
1210 POKE3,A1/256:POKE2,A1-PEEK(3)*256
1220 POKE5,A2/256:POKE4,A2-PEEK(5)*256
1230 POKE7,A3/256:POKE6,A3-PEEK(7)*256
1240 SYSA(3):RETURN
1250 REM ATTESA FIRE
1260 B$(1)-"PRESS FIRE":←GRAF0,1
1270 GOSUB1380:A-PEEK(56320):IFA<>111GOTO1270
1280 POKE53240,46:RETURN
1290 REM PROCEDURA DI MOVIE COMUNE A SAVE E COPY
1300 LU=0:Q=1
1310 SYSA(0),X1,Y1:SYSA(2),LX*8,LY,LD+LU
1320 ←CHAR 33,20,0,STR$(LD+LU)
1330 GOSUB1040:IFA<0THENGOSUB930:RETURN
```

```
1340 LU=0:FORA=0TOQ-1:LU=LU+NL(A):NEXT:GOTO1310
1350 REM CARICAMENTO SCRITTE
1360 RESTORE:FORA=0TON:READA$(A):←CHAR 34,A,0,A$(A):NEXT:RETURN
1370 REM CANCELLA SPAZIO PER MESSAGGI E SCRIVE I MESSAGGI
1380 A$="":FORC=1TO40:A$=A$+" ":NEXT:FORC=23TO24:←CHAR 0,C,0,A$:NEXT
1390 FORC=0TO1:←CHAR 0,23+C,1,B$(C):NEXT:RETURN
1400 REM PRENDE INFORMAZIONI DA JOY E LE SMISTA
1410 A=PEEK(56320):IFA=127GOTO1520
1420 A=A-116:IFA=<0ORA>10THENRETURN
1430 ONAGOTO1440,1460,1440,,1500,1480,1480,,1500,1460
1440 X=X+SP:IFA=1GOTO1500
1450 GOTO1520
1460 Y=Y+SP:IFA=2GOTO1440
1470 GOTO1520
1480 X=X-SP:IFA=6GOTO1460
1490 GOTO1520
1500 Y=Y-SP:IFA=5GOTO1480
1510 GOTO1520
1520 ON B GOTO 1540,1580
1530 RETURN
1540 ←COLOR0:←DRAW -160,VY,0,VX,VY,0:←DRAW VX,99,0,$,$,$:VX=X:VY=Y
1550 LX=(160+X)/8:LY=INT((100-Y)/8)
1560 GETA$:IFA$=CHR$(133)THENGOSUB1110
1570 ←COLOR1:←DRAW -160,Y,0,X,Y,0:←DRAW X,99,0,$,$,$:GOTO 1410
1580 ←PLOT VX+LX*8,VY,$:←PLOT VX,$,$:←PLOT $,VY+LY*8,$:←PLOT VX+LX*8,$,$
1590 VX=X:VY=Y
1600 ←PLOT X+LX*8,Y,$:←PLOT X,$,$:←PLOT $,Y+LY*8,$:←PLOT X+LX*8,$,$:GOTO1410
1610 DATA EDIT,MEMO,MOVIE,SAVE,LOAD,COPY,ALTRO,RESET
1620 DATA ------,DRAW,DELETE,INVERT,DROP,MOVE,RUN,CLEAR,EXIT
1630 GOSUB1360:Q=0
1640 REM CARICA LE LM ROUTINES
1650 READA$:IFVAL(A$)>255THENX=X+1:IFVAL(A$)<>CTHENPRINT"ERROR IN GRUPPO"X:END
1660 IFVAL(A$)>255THENC=0:GOTO1650
1670 IFA$="FINE"THENGOTO1750
1680 IFLEN(A$)>2THENA(Q)=LOC+W+1:Q=Q+1:PRINTA$,A(Q-1):GOTO1650
1690 X1=ASC(LEFT$(A$,1)):X2=ASC(RIGHT$(A$,1))
1700 IF X1>57THENX1=X1-55:GOTO1720
1710 X1=X1-48
1720 IFX2>57THENX2=X2-55:GOTO1740
1730 X2=X2-48
1740 W=W+1:Z=X1*16+X2:POKELOC+W,Z:C=C+Z:GOTO1650
1750 PRINT"FINAL LOC";LOC+W
1760 PRINT"PREMI UN TASTO":POKE198,0:WAIT198,1:RETURN
1770 REM
1780 DATA "INPUT DATI":REM GRUPPO 1
1790 DATA 20,F1,B7,86,04,20,F1,B7,86,05,A9,00,AA
1800 DATA 85,02,A9,E0,85,03,A5,05,C9,00,F0,13,18
1810 DATA E6,03,A9,40,65,02,85,02,90,02,E6,03,C6
1820 DATA 05,E4,05,D0,ED,A5,04,C9,00,F0,13,A9,08
1830 DATA 18,65,02,85,02,90,02,E6,03,C6,04,A5,04
1840 DATA C9,00,D0,ED,60,7370
1850 DATA "MEM FIGURA":REM GRUPPO 2
1860 DATA 20,F1,B7,86,07,20,F1,B7,86,08,20,FD,AE
1870 DATA 20,8A,AD,20,F7,B7,A5,14,85,05,A5,15,85
1880 DATA 06,20,5E,C5,20,58,C9,A5,08,85,F4,A9,00
1890 DATA AA,85,03,85,F3,A9,E0,85,04,A0,00,B1,03
1900 DATA C9,00,F0,49,A5,F3,C9,33,F0,31,A2,00,B1
1910 DATA 03,81,05,E6,05,A5,05,C9,00,D0,02,E6,06
1920 DATA C8,C4,07,D0,DF,E6,04,18,A9,40,65,03,85
1930 DATA 03,90,02,E6,04,C6,F4,A5,F4,C9,00,D0,C8
1940 DATA 20,6D,C5,20,60,C9,60,A9,00.85.F3.E6.05
```

```
1950 DATA A5,05,C9,00,D0,02,E6,06,88,18,90,C3,A5
1960 DATA F3,C9,33,F0,17,84,02,A0,00,98,91,05,C8
1970 DATA 91,05,C8,91,05,A9,33,85,F3,A4,02,88,18
1980 DATA 90,A6,84,02,A0,00,B1,05,AA,E8,8A,91,05
1990 DATA C9,00,D0,08,C8,B1,05,AA,E8,8A,91,05,A4
2000 DATA 02,18,90,94,21865
2010 DATA "RICHIAMA FIGURA":REM GRUPPO 3
2020 DATA 20,F1,B7,86,07,20,F1,B7,86,08,20,FD,AE
2030 DATA 20,8A,AD,20,F7,B7,A5,14,85,04,A5,15,85
2040 DATA 05,20,5E,C5,20,58,C9,A5,08,85,F4,A9,00
2050 DATA 85,F3,A0,00,A5,F3,C9,33,F0,37,A2,00
2060 DATA A1,04,C9,00,F0,5E,EA,EA:REM SOSTITUIBILI CON  11,02
2070 DATA 91,02,E6,04,A5,04,C9,00,D0,02,E6,05,C8
2080 DATA C4,07,D0,DF,E6,03,18,A9,40,65,02,85,02
2090 DATA 90,02,E6,03,C6,F4,A5,F4,C9,00,D0,C8,20
2100 DATA 6D,C5,20,60,C9,60,C6,09,A5,09,C9,00,D0
2110 DATA 02,C6,0A,A5,09,C9,FF,D0,18,A5,0A,C9,FF
2120 DATA D0,12,A9,00,85,F3,E6,04,A5,04,C9,00,D0
2130 DATA 02,E6,05,88,18,90,AD,A9,00,91,02:REM 91,02 SOSTITUIBILI CON EA,EA
2140 DATA 18,90,B0,A9,33,85,F3,E6,04,A5,04,C9,00
2150 DATA D0,02,E6,05,84,06,A0,00,B1,04,85,09,C8
2160 DATA B1,04,85,0A,A4,06,18,90,B1,21404
2170 DATA"TRANSFER MEM":REM GRUPPO 4
2180 DATA 20,5E,C5,A0,00,A2,00,B1,02,91,04,C8,C4
2190 DATA 06,D0,08,E4,07,D0,04,20,6D,C5,60,C0,00
2200 DATA D0,EB,E8,E6,03,E6,05,18,90,E3,4202,"FINE"
```

IN EDICOLA

IN EDICOLA
Commodore
COMPUTER CLUB
La rivista degli utenti di sistemi Commodore
L. 15.000
SPECIALE
UTILITIES
Hi-Res
Autorun
Turbo
Disk Drive
RAM-Disk
graphics
Files
Crunch
• DATA MAKER • SHUT OFF
• 4WD-RAMDISK • KOALA READER
• ASSEMBLER • DISK MERGER • TURBOKIT
• DISK MANAGER • AUTORUN MAKER • CRUNCHER
• TYPE • FORMAT SAVER • CROSS REFERENCE
• GRAPH COMPILER • RENUMBER • DELTAFIGHTER
in omaggio
IL SUPERGAME
DELTAFIGHTER
Systems

NON TUTTO, MA DI TUTTO

Piccola enciclopedia popolare sul C/16 ed il Plus/4

Di **Cynus**

Il Plus/4 ed il C/16 sono macchine dotate di potenzialità notevoli, ma di poco carburante, a differenza del C/64 il quale si avvale di distributori sparsi ovunque.

In queste pagine illustreremo alcuni trucchi per la programmazione delle macchine stesse ma, soprattutto, un metodo per trovarne altri per conto proprio.

Un listato invisibile

Provate a digitare quanto segue:

10 Rem linea 10
20 Rem linea 20
30 Rem linea 30

Dopo un List, modificate la linea 20 premendo i tasti riportati nelle parentesi qui di seguito:

20 REM (virgolette) (virgolette) (delete) (rvson) (aperta parentesi quadra) (rvsoff) (lettera O) (virgolette) (delete) (tasto cursore in alto) (return)

Se avete seguito alla lettera i suggerimenti, dovreste ottenere quattro caratteri dopo REM: le virgolette, una quadra aperta in campo inverso, la vocale "O" ed il pallino in campo inverso.

Se, però, chiedete List 20, la linea 20 sembra non esistere più; i nostri lettori più esperti avranno intuito la complicità della linea numero 30, che la copre interamente.

L'elaboratore si comporta in un modo così insolito perchè la procedura è attivata dallo stesso Sistema Operativo (S.O.) del computer.

E', infatti, a causa di questo che se si preme il tasto "cursore in alto" (d'ora in poi: "crsr up")il successivo carattere da evidenziare verrà visualizzato nella riga immediatamente superiore.

In generale, quindi, se si inserisce tale elemento nel listato, la linea successiva verrà listata una riga più in alto del dovuto; ciò non si verifica, ovviamente, riversando il listato su carta.

Se, però, si pone "crsr up" così com'è, il S.O. (sempre lui...) lo detokenizza, cioè ne converte il codice nella relativa parola basic, se esiste; nel caso di "crsrs up", manco a dirlo, un significato ad esso associato esiste davvero, con la conseguenza vista.

Altre utilissime funzioni sono attivabili, come è noto, premendo il tasto "Esc" seguito dalla pressione del tasto con la lettera "O": annullamento del modo virgolette, inversione, lampeg-

giamento ed inserimento.

Ciò consente, in modo virgolette, di evitare litigi col cursore che invece di rispondere ai tasti che lo dovrebbero spostare in lungo ed in largo per lo schermo, altera la linea ove si trova con strani simboli in reverse; semplicemente si premono uno dopo l'altro i tasti Esc ed O e si è subito fuori dalla lotta...

Questa caratteristica è proprio quella che viene impiegata nella riga 20: la parentesi quadra aperta (in reverse) equivale al tasto Esc, mentre la "O" rappresenta... se stessa.

I più avranno quindi intuito che sostituendo il simbolo di "crsr up" con altri caratteri speciali, si potranno ottenere gli effetti che fanno al caso proprio; ne suggeriamo solamente uno: ponete i simboli dei colori (tasto Commodore o Control con un numero da 1 a 8); si otterrà un listato in... multicolor.

Per non sbagliare, pertanto, adoperate il trucchetto per nascondere ad occhi indiscreti (che non siano lettori di C.C.C.) le linee che credete, con una sola raccomandazione: la linea sottostante deve essere lunga almeno quanto quella contenente la REM, in modo da coprirla per intero.

Disassemblare su carta

Veniamo ora al problema di riversare, su carta, i programmi in L.M. servendosi del Monitor incorporato.

L'operazione più banale, di solito, consiste nell'impartire...

OPEN 4,4:CMD 4

...e, subito dopo...

MONITOR

...ma il foglio, in tal modo, viene deturpato dai contenuti dei registri del 7501 che, anzichè andare sul video, vengono dirottati su stampante, esattamente come il CMD 4 ha ordinato al S.O.

Anche qui la soluzione è banale ed altrettanto efficace; vediamo un modo per accostarsi al problema.

Prima del comando CMD4 il foglio è ancora pulito; si sporca solo entrando in Monitor: è a questo punto, pertanto, che bisogna cambiare qualcosa.

Il buon S.O. traduce ogni comando basic (e quindi anche Monitor) in una serie di JSR (acronimo inglese di "Jump to Sub-Routine", equivalente L.M. di GOSUB) che interessano varie sottoprocedure che svolgono in parte o del tutto quanto richiesto dal comando stesso.

Nel caso di Monitor il primo JSR è al comando R del Tedmon che, appunto, visualizza i famigerati registri.

Siccome il comando Monitor (e pure gli altri) non è modificabile essendo su ROM, possiamo però fare iniziare il comando a partire dalla JSR, oppure dalla parte in L.M. del comando che più ci aggrada.

Disassemblando la ROM, si trova facilmente(!) che il comando in questione inizia dalla locazione 62533 (esadecimale $F445); se, quindi, dopo il comando CMD 4 si impartisce una SYS a tale locazione, anzichè il comando Monitor, si ha a disposizione un comando "modificabile".

Riscrutando nella memoria si trova che l'ultima JSR di visualizzazione messaggi è alla locazione 62613 ($F495); dal momento che l'istruzione JSR occupa tre bytes in memoria (uno per il codice "JSR", due per l'indirizzo della subroutine stessa), ne consegue che la prima locazione del nostro comando Monitor sarà la 62616 ($F498).

Ecco, quindi, la sequenza dei comandi da impiegare:

Open 4,4: Cmd 4: Sys 62616

Al momento del Return vi troverete in ambiente Tedmon anche se sembra che non sia accaduto nulla; ad ogni successiva pressione dello stesso tasto si ottiene su carta l'output del comando impartito: generalmente M (dump Memory) oppure D (Disassemble).

Quando è stato stampato l'ultimo carattere voluto, la sequenza: X + (Return) + Close 4 + Print#4 + (Return) chiuderà correttamente il file senza aggiungere altri caratteri allo stampato.

Come si nota, quindi, la maggior parte dei problemi di programmazione, ha la sua soluzione in qualche locazione; la questione non è tanto il trovarla, quanto il saperla impiegare correttamente.

Parlando del cursore

Molte mappe di memoria per C/16 & Plus/4 riportano che i registri numero 12 e 13 del Ted Chip (rispettivamente le locazioni 65292 esa $FF0C e 65293 esa $FF0D), contengono la posizione del cursore sullo schermo.

E' anche noto che il video contiene mille (25 righe x 40 colonne) diverse caselle ove un carattere (e quindi anche il cursore che altro non è se non uno spazio in reverse) può essere visualizzato; la posizione, quindi, può andare da 0 (posizione Home, in alto a sinistra) a 999 (ultima cella in basso a destra).

Nella rappresentazione binaria, per rappresentare tale numero, sono necessari almeno 10 bits e, siccome il 7501 tratta valori ad 8 bits (il famoso byte), si è reso necessario l'impiego di 2 byte (cioè 16 bits) per svolgere tale funzione; è chiaro, però, che in 2 byte possiamo immettere valori maggiori di 999, sino a 65535; solo che quelli superiori a 999 non visualizzeranno il cursore (durante l'esecuzione di un programma, per esempio), anche perchè i 6 bits più alti di 65292 non fanno parte del registro di posizione cursore ma sono, per così dire inutilizzati.

A qualcuno la faccenda potrà anche non interessare sino a quando non debba, ad esempio, simulare un INPUT usando GETKEY.

Un'applicazione di quanto visto sopra può essere il seguente programmino:

```
10 rem getput
20 :
30 scnclr
40 print "scelta? ";
50 x=peek(202): y=peek(205): gosub 100
60 :
70 getkey a$: a=asc(a$): if a<49 or a>57 then 70
80 print a$: end
90 :
100 k=y*40+x: h=int(k/256): l=k-h*256
110 poke 65292,h: poke 65293,l: return
```

Nell'esempio si è limitata la scelta ad un valore compreso tra 1 (asc=49) e 9 (asc=57) ed è stato dato ad X (riga)

ed Y (colonna) il valore corrente della posizione del cursore che, dopo la Print di linea 40, si troverà subito dopo l'ultimo carattere stampato, in modo da far lampeggiare il cursore subito dopo il carattere di spazio.

Nella routine, K rappresenta il valore da 0 a 999 che viene scomposto nella parte alta (H=High) e nella parte bassa (L=Low) che andranno pokate rispettivamente nei registri 65292 e 65293.

Il registratore

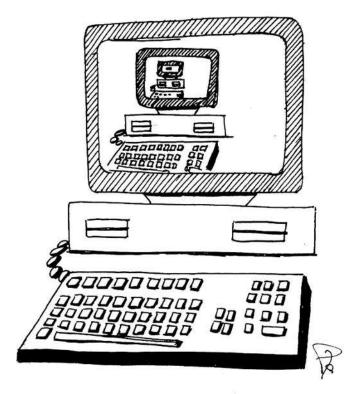
Con la stessa ottica di impiego, si può leggere, in alcuni testi, che il buffer di cassetta si trova nelle locazioni da 819 a 1010, ma la notizia è piuttosto vaga e di scarsa utilità pratica... o no?

Frugando nelle suddette locazioni, prima, durante e dopo un caricamento da cassetta, si trova quanto segue:

819-820 Indirizzo iniziale (basso-alto)
821-822 Indirizzo finale (basso-alto)
823-838 Nome del programma (16 caratteri)
839-1010 Dati del programma

Sfruttando il contenuto di queste prime quattro locazioni si ricavano queste cinque linee:

```
10 s=819
20 open 1,1,0
30 print "inizio:" peek(s) + 256*
peek(s+1)
40 print "fine:" peek(s+2) + 256*
peek(s+3)
50 Close1
```

La routine visualizza le locazioni di memoria dove un programma (basic o l.m.) inizia e finisce; cioè, in pratica, dove verrà caricato il programma stesso; è impiegabile anche sugli altri computer Commodore, sostituendo il valore di S con quello di inizio del buffer di cassetta: per il VIC 20 ed il C/64 S=828; per i C/128 S=2816.

La compatibilità

Una delle necessità di un programmatore è quella di universalizzare i programmi in modo che possano girare su qualunque macchina.

Alle volte, infatti, è indispensabile dirottare il programma verso una routine piuttosto che un'altra, a seconda dell'elaboratore sul quale il programma sta girando; ciò si realizza con l'assegnamento di una variabile dal cui contenuto si desume il tipo di computer; ad esempio:

If cpu=128 then... (istruzioni per C/128)
If cpu=64 then... (istruzioni per C/64)

La variabile, ovviamente, deve essere assegnata all'inizio del programma, ed ecco il punto: come si può fare?

Una delle possibili soluzioni può essere la locazione 61654 che ha i seguenti contenuti:

240 (Commodore PET Basic 1.0)
169 (PET Basic 2.0)
32 (PET Basic 4.0)
145 (VIC 20)
255 (C/16 & Plus/4)
82 (C/64)
0 (C/128)

Se, addirittura, si avesse la necessità di distinguere tra il C/64 "originale" e quello "contenuto" nel C/128, si faccia riferimento (dopo aver accertato che PEEK(61654)=82!) alla locazione 1, che deve contenere:

55 (C/64 originale)
119 (C/128 in modo 64)

Colori

Nel numero 44 di C.C.C. si è parlato a proposito della locazione 134 che contiene il colore principale, con la quale non è però possibile modificare il colore del cursore mediante una poke.

La locazione corretta da impiegare a tal scopo è la 1339 che contiene il byte attivo di attributo cioè, più semplicemente, un valore che rappresenta il colore, la luminosità e l'eventuale lampeggiamento coi quali visualizzare i caratteri ed il cursore.

I valori da pokare per ottenere quan-

to voluto, vanno calcolati come segue:

C = colore desiderato (da 1 a 16)
L = luminosità desiderata (da 0 a 7)
*Valore da pokare = (L*16) + (C-1)*

Il numero ottenuto va aumentato di 128 se si vuole anche il lampeggiamento.

Sempre a proposito di colori, ecco un listato monolinea da impiegarsi nelle proprie creazioni in basic 3.5...

*10 Input c,l: poke 274+c,l*16+c-1*

...che è nato dalla constatazione che la tabella luminosità e dei colori si trova in RAM dalla locazione 275 alla 289: ma che significa?

Grande è la comodità di cambiare il colore del cursore premendo il tasto Shift o Commodore, insieme ad un numero da 1 a 8; con il tasto 4, ad esempio, il cursore diventa rispettivamente azzurro (tasto Shift) o rosa (Commodore).

Se qualche colore ha la luminosità che non ci soddisfa, la possiamo modificare a piacimento con la monoriga vista sopra: vediamo come.

Si inseriscono colore (C) e luminosità (L) separati da una virgola ed il programmino fa il resto; calcola il valore da pokare con una formula già vista: valore = (L*16)+(C-1).

Calcola anche la locazione dove pokare il tutto, tenendo presente che la 275 contiene il valore relativo al colore 1 (Nero), la 276 al 2 (Bianco) etc. sino alla 289 relativa al 16 (Verde chiaro).

Ecco quindi che un'applicazione pratica può essere quella di inserire i simboli in reverse dei colori all'interno di un comando Print anzichè con il solito Color 1,x,y.

Alcune routine S.O.

Detto questo vediamo di scoprire come si possono adoperare varie routine del S.O. per ciò che più aggrada.

Quanto segue è impiegabile da basic o, ancor meglio, in programmi Assembly; vediamo un primo esempio.

Per stampare un numero da 0 a 65535, è molto veloce e pratico impiegare la routine del S.O. (che parte da 42067 $A453) che stampa la parola "IN", seguita dal numero di linea basic.

Bisogna evitare la prima parte di tale routine ed assumere, come punto di ingresso (entry point) l'indirizzo 42079 ($A45F) caricando dapprima l'accumulatore con la parte alta ed il registro X con quella bassa:

LDA #$ high
LDX #$ low
JSR $A45F

Da basic si può provare con:

Poke 2034, high: Poke 2035, low: Sys 42079

Ecco ora una considerazione importante: sapendo quali locazioni impiega una qualunque procedura del S.O. (comando basic, funzione di screen editor etc.) è possibile modificare quelle che si trovano in RAM per alterare la normale procedura ed ottenerne un'altra simile, diversa, opposta etc.

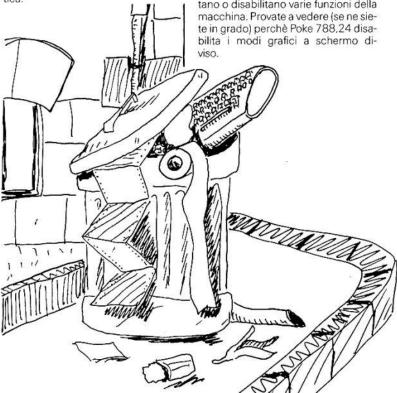
Consideriamo i cosiddetti vettori delle varie funzioni basic LOAD, SAVE, OPEN ed altre: ponendo valori differenti si indirizzerà il S.O. verso una qualunque routine, nostra o del S.O. stesso; vediamo quanto detto in pratica.

Uno dei vettori più elementari è quello della routine d'interrupt che nel C/16 e Plus/4 si trova in 788-789 e punta a $CE0E ove, appunto, inizia tale routine; disassemblandola si nota che le prime tre istruzioni controllano se ci si trova in modalità grafica 2 oppure 4, nelle quali, come è noto, lo schermo deve essere suddiviso nelle due aree grafica + testo.

Se il controllo è positivo viene eseguita la routine di split screen che, appunto (via raster) provvede a lasciare in modo testo le ultime cinque righe di caratteri; se invece è negativo, salta a $CE18 SENZA fare lo split screen.

Se (esempio puramente didattico...) voglio inibire tale caratteristica, basterà inserire in 788-789 l'indirizzo esa $CE18 nella solita forma low/high; CE è la parte alta mentre 18 è quella bassa; in decimale $CE=206, $18=24: le due corrispondenti poke saranno quindi:

Poke 788,24: poke 789,206

Siccome la locazione 789 contiene già 206 (lo standard è $CE0E), sarà sufficiente solo la prima poke; ecco quindi con quale criterio si scoprono (e si valutano...) le varie poke che abilitano o disabilitano varie funzioni della macchina. Provate a vedere (se ne siete in grado) perchè Poke 788,24 disabilita i modi grafici a schermo diviso.

Reset ed Antireset

Ecco ora le Sys di reset dei vari Commodore e dove si trovano: in tutti i computers basati su un processore della famiglia 6502, l'indirizzo di reset è SEMPRE contenuto nelle locazioni 65532 ($FFFC) e 65533 ($FFFD) e ne consegue che Print PEEK(65533)* 256 + PEEK(65532) fornisce le sys di reset:

Sys 64802 (VIC 20)
Sys 65529 (C/16-Plus/4)
Sys 64738 (C 64)
Sys 65341 (C/128)

Ed ecco, al contrario, le righe da digitare dopo un comando New o la pressione accidentale(!) del tasto di reset, allo scopo di recuperare il programma basic in memoria.

Per il VIC 20 inespanso:
Poke 4098,16: Sys 50483: Poke 45, Peek(34): Poke 46, Peek(35): Clr

Per il VIC 20 con espansione:
Poke 4610,18: Sys 50483: Poke 45, Peek(34): Poke 46, Peek(35): Clr

Per il C/16-Plus/4:
Poke 4098,16: Sys 34840: Poke 45, Peek(34): Poke 46, Peek(35): Clr

Per il C/64:
Poke 2050,8: Sys 42291: Poke 45, Peek(34): Poke 46, Peek(35): Clr

Per il C/128:
Poke 7170, 28: Sys 20303: Poke 47, Peek(36): Poke 48, Peek(37): Clr

Queste poche istruzioni non fanno altro che rimettere i valori corretti nelle locazioni alterate dal NEW o dal reset; utilizzano la routine del S.O. di relink che, appunto, ristabilisce i corretti puntatori, depositandoli però in locazioni di "parcheggio" da dove le ultime due poke le pongono al posto corretto.
Per concludere, provate queste due Sys:

C/16 Plus/4: Sys 52651
C/128: Sys 32800,123,45,6

A che servono? Scopritelo da soli...

Una manciata di Poke

Indichiamo un gruppo di valori da tener presente per i vostri programmi:

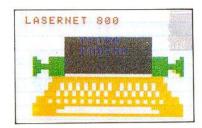
Poke 0,0 :isola il computer da tutti i dispositivi esterni
Poke 22,35 :elimina dal LIST i numeri di linea
Poke 239,0 :pulisce il buffer di tastiera, accettando solo i tasti premuti dopo la poke stessa; una sua tipica applicazione è individuabile prima di un GETKEY
Poke 768,0 :modifica il vettore Ierror, provocando un break ed impedendo il ritorno in basic
Poke 770,3 :modifica il vettore di warm start facendolo ripuntare a se stesso: stampa continuamente READY.
Poke 770,6 :stesso vettore che punta a se stesso ma senza passare per la routine di ready: il computer si blocca
Poke 772,0 :modifica il vettore di tokenizzazione facendolo puntare ad una JSR per l'emissione del ?SYNTAX ERROR all'introduzione di un comando
Poke 772,108 :stesso vettore: il comando non verrà considerato; con il Plus/4 si avrà, in più, un messaggio ben conosciuto
Poke 774,2 :vettore di LIST: lista la prima linea in continuazione
Poke 774,92 :stesso vettore: lista solo i numeri di linea
Poke 774,107 :stesso vettore: punta a se stesso e blocca il tutto
Poke 774,196 :stesso vettore: punta alla routine di run causando l'omonimo comando se il programma inizia con la linea numero zero
Poke 774,214 :stesso vettore che punta alla routine di ?SYNTAX ERROR
Poke 774,246: Poke 775,255 :vettore che punta alla routine di reset
Poke 774,123: Poke 775,138 :vettore che punta alla routine di NEW
Poke 786,69 :secondo vettore di Interrupt: la routine di IRQ non verifica più la pressione del tasto Run/Stop e non aggiorna le variabili TI e TI$
Poke 786,72 :come sopra; in più renderà inutile il terzo parametro del comando "Sound": i suoni avranno durata infinita
Poke 786,93 :come sopra; in più non verrà più eseguita la scansione di tastiera, disabilitando pertanto tale dispositivo
Poke 806,114 :impedisce di verificare la pressione del tasto Run/Stop,
Poke 814,239 :vettore di LOAD: ne impedisce il comando relativo
Poke 816,136 :vettore di SAVE: risponde con ?OUT OF MEMORY
Poke 816,168 :stesso vettore: risponde con ?ILLEGAL DEVICE NUMBER
Poke 816,205 :stesso vettore: disabilita LOAD & SAVE
Poke 816,255 :stesso vettore: risponde con ?BREAK ERROR
Poke 817,255 :stesso vettore (byte alto): il SAVE non verrà più considerato
Poke 1144,0 :modifica la routine Chrget: il computer non accetta i comandi
Poke 1343,0 :pone a zero la grandezza del buffer di tastiera, cioè non accetta nessun tasto disabilitando in pratica la tastiera
Poke 1344,0 :l'autorepeat funziona solo con i tasti cursore, spazio e delete
Poke 1344,64 :l'autorepeat è disabilitato per tutti i tasti
Poke 2025,128 :disabilita lo scroll dello schermo; equivale alla funzione di screen editor Esc + M
Poke 65286,11 :incrementa la velocità di elaborazione di circa il 33% disabilitando lo schermo.

PRONTO?... IL SOFTWARE E' IN LINEA

Una novità di sicuro interesse per gli appassionati di telematica

di **Michele Maggi**

Il grande successo dell'adattatore telematico per C/64-128 ha rappresentato una novità per gli hobbisti che, già da tempo, lamentavano la mancanza di nuovi prodotti dedicati ai "piccoli" Commodore.

La telematica ha senz'altro aperto nuove vie allo scambio "elettronico" di dati e, perchè no?, anche di

software tra hobbisti dislocati nelle più diverse zone d'Italia.

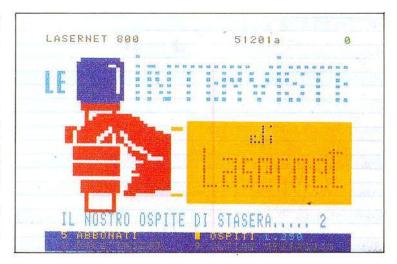
Inutile dire che il modem, da solo, non serve (quasi) a nulla: senza il relativo complemento, espresso come rete o network, non ha infatti funzione alcuna.

Fortunatamnte, allegato al pacchetto contenente l'Adattatore Tele-

matico, c'è anche la password che consente, per un anno intero, l'accesso al servizio Videotel ed alle Pagine Gialle Elettroniche (PGE).

I servizi Lasernet

Tra le varie Banche Dati che operano tramite Videotel, particolarmente interesaante è il servizio telematico "Lasernet 800" perchè offre una pluralità di servizi sicuramente interessanti per gli utenti di home computer e, in particolare, per i sessantaquattristi.

I servizi offerti da Lasernet 800 si riconducono a tre categorie principali: Tele-software, Home Entertainment e Communications.

La telematica non è un'opinione

Il boom della telematica è iniziato circa tre anni fa con la comparsa sul mercato di vari tipi di modem dedicati al C/64.

Il successo, però, non fu immediato, non tanto perchè i prodotti non fossero affidabili, ma perchè da una parte l'utente non era ancora preparato per il "salto" verso la telematica; dall'altra si sentiva la mancanza di network o banche dati che potessero offrire ai neopossessori di modem le soddisfazioni giustamente richieste.

I servizi tipo Videotel e Lasernet svolgono, appunto, tale funzione, senza dimenticare la possibilità di arricchire la propria biblioteca software a costi bassissimi.

Una recentissima novità riguarda da vicino i nostri lettori: da questo mese è possibile procurarsi, viaLasernet, gran parte del software della Systems Editoriale, tramite l'accesso al Box "Systems Editoriale" contenuto nei menù Lasernet.

Tutto il software messo a disposizione dalla Systems è dedicato ad utenti di C/64-128, C/16 Plus/4, Vic 20, Spectrum ed MSX.

La prima categoria di servizi offre un elenco di programmi trasferibili, naturalmente via telefono, su cassetta (o su disco); la seconda, Home Entertainment, consiste in una sere di rubriche che hanno la funzione di intrattenere l'utente senza, però, richiedergli eccessivo impegno.

Le tipiche applicazioni di Home Entertainment sono "I corsi di Lasernet", dedicati a chi voglia apprendere il Basic oppure il Linguaggio Macchina del proprio computer, e notizie e trucchi per tutti gli smanettoni.

La terza categoria, "Communications", facilita l'interazione con gli altri utenti, o con il sistema Lasernet stesso.

La sezione più interessante di "Communications" è senz'altro la possibilità di effettuare interviste in diretta a personaggi che ricoprono importanti ruoli nel mondo dell'informatica.

I costi per accedere al servizio Lasernet sono molto bassi e quindi alla portata di chiunque.

Per maggiori informazioni:
Lasernet 800
Via G. Modena, 9
20129 Milano
Tel. 02/20.02.01

Commodore

Prima di tutto

Ancora una volta il nome Commodore risulta essere sia tra i primissimi posti nella graduatoria di apparecchi venduti (1987) che nelle previsioni del primo semestre di quest'anno.

L'inchiesta, recentemente conclusa, conferma che l'andamento delle vendite è stato superiore alle previsioni, perfino per i modelli (come il PC.I) presentati al pubblico quasi al termine dell'anno scorso.

Negli Stati Uniti l'Amiga rappresenta ormai il 40% del fatturato globale Commodore.

80386

Anche nel mercato dei professionali 386 la Commodore ha voluto esser presente proponendo una configurazione base operante a 6/16 Mhz, dotata di 2 mega Ram.

E' disponibile un disco rigido di 40 oppure 80 mega con mouse e Windows-386 di serie (per il modello Pc 60/80). Di serie anche la scheda grafica Ega ed un monitor monocromatico.

Software Amiga

E' ormai operante l'accordo tra Commodore Italiana e la C.T.O. di Bologna (Via dell'Indipendenza, 40) in base al quale l'azienda si impegna a fornire, a prezzi contenuti, software professionale (e non) per Amiga corredato di istruzioni in italiano.

La maggior parte dei giochi, comunque, è proposta in lingua inglese, grazie alla notevole facilità di gestione intuitiva del game.

Alcuni esempi di prezzi: videogame da L.27000 a L.38000; Pagesetter, il favoloso Desk Top Publishing, L.210000; Logistix, L.120000. I prezzi comprendono anche l'I.V.A.

Nuovi accessori

E' iniziata la commercializzazione del nuovo disk drive 1541 II, che differisce dal precedente 1541 solo per l'alimentazione esterna: le funzionalità (e la compatibilità) rimangono infatti rigorosamente identiche.

Il monitor a colori 2080, del tipo a lunga persistenza, dovrebbe accontentare qualsiasi utente Commodore, sia che usi il C/64, sia che preferisca l'Amiga, sia che desideri sfruttarlo con il PC I.

Amiga per tutti

La divisione commerciale della Commodore Italiana, sollecitata da un mercato particolarmente favorevole, sta studiando forme di vendita vantaggiose per coloro che desiderano procurarsi un Amiga.

Maggiori dettagli sull'iniziativa saranno divulgati, entro la primavera, dai Commodore Point, punti di riferimento privilegiati per lo sviluppo di una simile attività promozionale.

Amiga alla RAI

In orario notturno (dalle 23:30 alle 3 del mattino) Amiga verrà utilizzato in una trasmissione su RAI 3, il sabato, per sfruttarne le potenzialità grafiche e sonore.

La trasmissione, iniziata il 15 febbraio, dovrebbe protrarsi fino alla fine di giugno e sarà ambientata in spazi tecnologicamente evoluti (c'era da dubitarlo?)

GRAFICA IN MEDIA RISOLUZIONE

Un listato utile per chi intenda approfondire le proprie conoscenze sul linguaggio macchina

di **Fabio Budai**

Nonostante l'argomento sia stato già affrontato in precedenza sulle pagine della nostra Rivista, pubblichiamo volentieri l'articolo del nostro lettore per una serie di motivi.

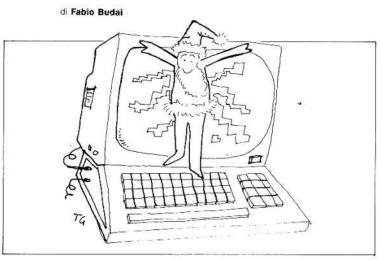
Prima di tutto c'è da rilevare che un disassemblato commentato è sempre utile per impadronirsi di varie tecniche di programmazione ed il contributo di chiunque si rileva prezioso; inoltre i lettori, per esercitarsi nell'affascinante mondo dell'Assembly, potrebbero modificare il programma in modo da eliminare l'incongruenza dello scambio righe/colonne.

Più interessante, sempre a livello di intervento sul listato pubblicato, è sicuramente lo studio per inserire una routine di controllo che impedisca di accettare valori che escano dal range imposto (50x80) mediante l'emissione di un "illegal quantity error".

La routine, infatti, si blocca nel caso in cui si tenti di assegnare valori eccessivi agli argomenti delle Sys, e non è possibile recuperare l'uso della tastiera.

La routine, divisa in due parti, permette di scrivere, cancellare e esaminare un punto sullo schermo gestito in modo testo.

Il programma, infatti, considera quest'ultimo come una matrice di 80x50 punti e ciò è possibile grazie al ricorso ai caratteri grafici del C/64 che, opportunamente gestiti, consentono la riproduzione di immagini di notevole efficacia.

Naturalmente non è possibile la sovrapposizione dei punti con il testo (come invece avviene usando l'hi-res) ma, con un po' di attenzione, è possibile usare contemporaneamente testo e punti.

I parametri da passare alla routine sono tre, nel caso si voglia scrivere un punto, due negli altri casi.

I primi due, X e Y, rappresentano le coordinate del punto e possono variare fra fra 0 e 79 (asse Y) e tra 0 e 49 per ciò riguarda l'asse X che viene qui considerato verticale; una mia distrazione in fase di programmazione, purtroppo, ha causato questo piccolo disguido: normalmente, come è noto, X si riferisce alla colonna, mentre nel programma di queste pagine rappresenta la riga di schermo; e viceversa per la Y.

Il terzo parametro, relativo al colore, può avere i soliti valori usati per il colore del video.

La sintassi dei comandi

La gestione dei 4000 punti indirizzabili è piuttosto semplice e si realizza mediante tre forme sintattiche; la prima permette di tracciare un punto, la seconda di cancellarlo e la terza di esaminarne l'eventuale presenza:

SYS 40000,X,Y,C
SYS 40006,X,Y
SYS 40003,X,Y

Il risultato della terza forma sintattica (-1 se il punto è spento e un numero compreso fra 0 e 15, indicante il colore, negli altri casi) viene riportato in ST.

Un'ultima considerazione riguarda il colore (parametro C) che, almeno all'inizio, può creare problemi. Infatti i punti settabili sono 4000, mentre le celle colore sono 1000. Ne consegue che i punti di coordinate pari hanno lo stesso colore di quelli dispari precedenti e ciò sia in verticale che in orrizzontale.

SCHEDA TECNICA

Software didattico per applicazioni grafiche

Hardware richiesto: C/64: difficilmente adattabile ad altri computer Commodore

Ideale l'uso di un monitor a colori

Consigliato a tutti gli appassionati di grafica ed a coloro che vogliano cimentarsi con il linguaggio macchina.

Anche il programma pubblicato in queste pagine è contenuto nel disco "Directory" di questo mese.

```
  1 POKE 55,63:POKE 56,156:PRIN
    T"[CLEAR][2 DOWN][4 RIGHT]S
    CRITTURA DATI LOW PLOT (R/W
    /C)"

  5 FOR A=40000 TO 40263:READ D
    :POKE A,D:B=B+D:NEXT:IF B<>
    30512 THEN PRINT,"[2 DOWN]E
    RRORE":END

 10 PRINT"[4 DOWN]RUTINE ATTIVA
    TE"
 11 PRINT"[DOWN]SYS40000,Y,X,C
    SCRIVE"

 12 PRINT"SYS40003,Y,X TEST:RIS
    ULTATO(-1/15)IN ST"
 13 PRINT"SYS40006,Y,X CANCELLA
    "

 14 :
100 DATA  76,166,156,76,221,156
    ,76,17,157,56,32,240,255,14
    2,55

101 DATA  3,140,56,3,160,0,173,
    52,3,74,144,1,200,141,52
102 DATA  3,173,53,3,74,144,2,2
    00,200,141,53,3,185,146,156

103 DATA  141,57,3,174,52,3,172
    ,53,3,24,32,240,255,152,101
104 DATA  209,144,2,230,210,133
    ,209,162,15,160,0,177,209,2
    21,149

105 DATA  156,240,3,202,208,248
    ,96,1,4,2,8,126,124,226,123
106 DATA  97,255,236,108,127,22
    5,251,98,252,254,160,0,32,2
    53,174

107 DATA  32,158,183,142,52,3,3
    2,253,174,32,158,183,142,53
    ,3

108 DATA  32,253,174,32,158,183
    ,142,54,3,32,73,156,138,13,
    57
109 DATA  3,170,189,149,156,145
    ,209,24,165,210,105,212,133
    ,210,173

110 DATA  54,3,145,209,76,62,15
    7,32,253,174,32,158,183,142
    ,52

111 DATA  3,32,253,174,32,158,1
    83,142,53,3,32,73,156,138,4
    5
112 DATA  57,3,141,54,3,24,165,
    210,105,212,133,210,177,209
    ,41

113 DATA  15,160,255,174,54,3,2
    40,1,168,132,144,76,62,157,
    32
114 DATA  253,174,32,158,183,14
    2,52,3,32,253,174,32,158,18
    3,142
115 DATA  53,3,32,73,156,142,54
    ,3,173,57,3,73,255,45,54

116 DATA  3,170,208,5,169,32,76
    ,60,157,189,149,156,145,209
    ,174
117 DATA  55,3,172,56,3,24,76,2
    40,255
118 END
```

```
100 REM DIMOSTRATIVO DI GRAFICA
     IN MEDIA RISOLUZOINE PER C
    /64

110 REM DI FABIO BUDAI PER C/64
120 :
130 PRINTCHR$(147):X$="

       ":REM 39 SPAZI
140 FOR I=-3.1415 TO 3.1415 STE
    P .13:X=SIN(I)*40+40:Y=Y+1
150 IF Y<50 AND X<80 THEN GOSUB
     170:NEXT

160 END
170 SYS40000,Y,X,1:PRINTCHR$(19
    )X$;CHR$(19);CHR$(18)"X:"CH
    R$(146);

180 PRINTX;CHR$(18)"Y"CHR$(146)
    ;Y;SIN(I):RETURN
```

```
     Disassemblato commentato
   "Grafica in media risoluzione"
   ----------------------------

 9c40 jmp $9ca6   Setta un punto
 9c43 jmp $9cdd   Testa un punto
 9c46 jmp $9d11   Cancella un punto
```

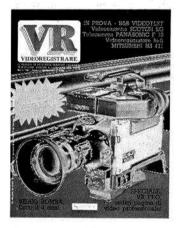
La parte seguente, comune alle tre funzioni, legge e calcola la posizione del punto.

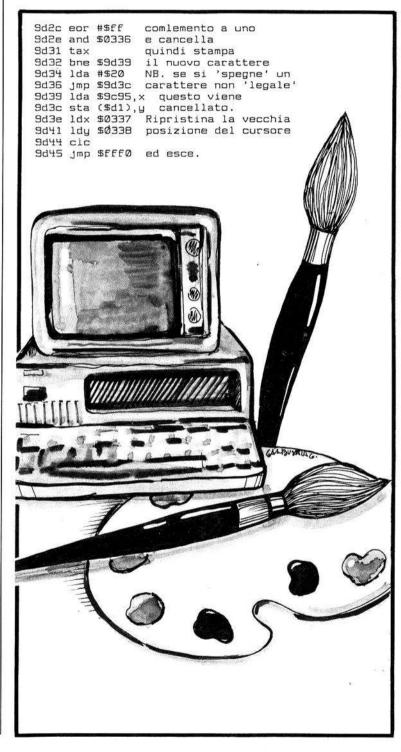
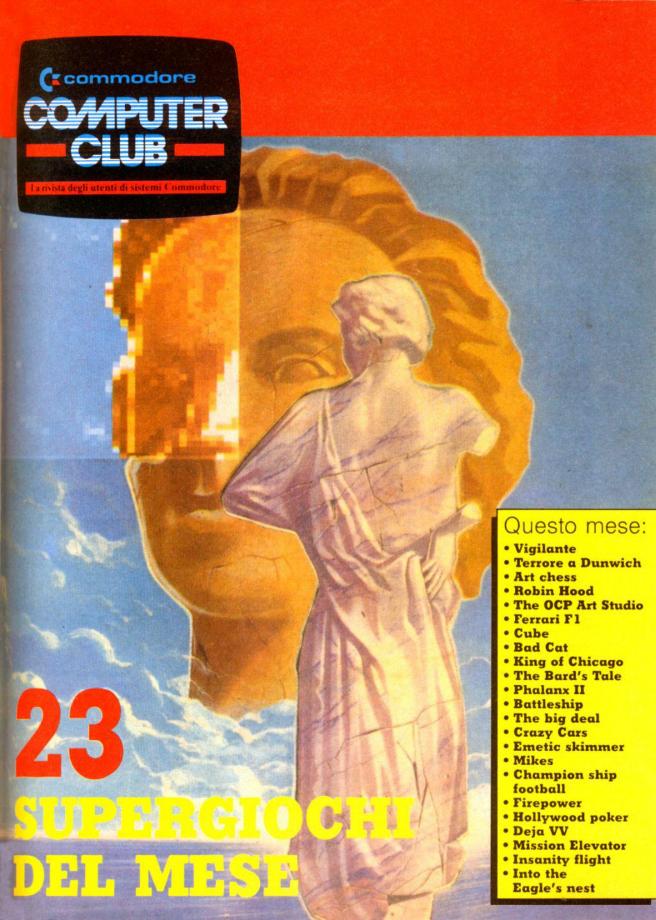
```
 9c49 sec           Legge le coord. del
 9c4a jsr $fff0     cursore col Kernal
 9c4d stx $0337     e la memorizza per poi
 9c50 sty $0338     ripristinarla
 9c53 ldy #$00      y fa da offset
 9c55 lda $0334     preleva la coord. X
 9c58 lsr a         e la divide per 2
 9c59 bcc $9c5c     c'e' resto?
 9c5b iny           Si, increm. l'offset
 9c5c sta $0334     memorizza la X
 9c5f lda $0335     preleva la Y
 9c62 lsr a         la divide per 2
 9c63 bcc $9c67     c'e' resto?
 9c65 iny           Si, increm. due volte
 9c66 iny           l'offset
 9c67 sta $0335     memorizza la Y
 9c6a lda $9c92,y    preleva la maschera
 9c6d sta $0339     e la memorizza
 9c70 ldx $0334     Legge le coord X e Y
 9c73 ldy $0335     calcolate e
 9c76 clc           posiziona il cursore
 9c77 jsr $fff0     usando il Kernal
 9c7a tya           aggiusta il puntatore
 9c7b adc $d1       di schermo aggiungendo
 9c7d bcc $9c81     il valore della
 9c7f inc $d2       coordinata Y
 9c81 sta $d1
 9c83 ldx #$0f      Viene poi confrontato
 9c85 ldy #$00      il carattere dello
 9c87 lda ($d1),y    schermo con quelli
 9c89 cmp $9c95,x    grafici ammessi
 9c8c beq $9c91     Finito il ciclo in x
 9c8e dex           ci sara' il numero
 9c8f bne $9c89     logico del carattere
 9c91 rts           Esce.

 9c92 01 04 02 08
                     Maschera per
                     l'individuazione
                     dell'esatta posizione
                     del punto all'interno
                     della cella video.
 9c96 7e 7c e2 7b 61 ff ec 6c
 9c9e 7f e1 fb 62 fc fe a0 00
                     I precedenti valori
                     sono i codici poke
                     dei caratteri grafici
                     che compongono i
                     punti.
```

```
Setta un punto
--------------
 9ca6 jsr $aefd    Salta la virgola
 9ca9 jsr $b79e    preleva la X
 9cac stx $0334    la memorizza
 9caf jsr $aefd    salta virgola
 9cb2 jsr $b79e    preleva la Y
 9cb5 stx $0335    la memorizza
 9cb8 jsr $aefd    salta la virgola
 9cbb jsr $b79e    preleva il colore
 9cbe stx $0336    lo memorizza
 9cc1 jsr $9c49    calcola la posizione.
 9cc4 txa          In a num. logico car.
 9cc5 ora $0339    or inclus.=punto on
 9cc8 tax          x fa da puntatore
 9cc9 lda $9c95,x   nella tabella
 9ccc sta ($d1),y   stampa carattere
 9cce clc          e,aggingendo alla
 9ccf lda $d2      parte alta del puntat.
 9cd1 adc #$d4     dello schermo $d4
 9cd3 sta $d2      viene colorato
 9cd5 lda $0336    il carattere
 9cd8 sta ($d1),y
 9cda jmp $9d3e    procedura di uscita.
```

```
Test su un punto
----------------
 9cdd jsr $aefd    Salta virgola
 9ce0 jsr $b79e    preleva X
 9ce3 stx $0334    e la memorizza
 9ce6 jsr $aefd    salta virgola
 9ce9 jsr $b79e    preleva Y
 9cec stx $0335    e la memorizza
 9cef jsr $9c49    calcola la posizione
 9cf2 txa          in a num. logico car.
 9cf3 and $0339    confronta
 9cf6 sta $0336    memorizza il risultato
 9cf9 clc          legge il colore
 9cfa lda $d2
 9cfc adc #$d4
 9cfe sta $d2
 9d00 lda ($d1),y
 9d02 and #$0f     solo la parte bassa
 9d04 ldy #$ff     y=-1 (punto spento)
 9d06 ldx $0336    se x = 0
 9d09 beq $9d0c
 9d0b tay          altrimenti y=colore
 9d0c sty $90      mette in ST.
 9d0e jmp $9d3e    Procedura di uscita
```

```
Spegne un punto
---------------
 9d11 jsr $aefd    Salta virgola
 9d14 jsr $b79e    prende X
 9d17 stx $0334    la memorizza
 9d1a jsr $aefd    salta virgola
 9d1d jsr $b79e    prende y
 9d20 stx $0335    la memorizza
 9d23 jsr $9c49    calcola posizione
 9d26 stx $0336    memorizza num. logico
 9d29 lda $0339    preleva la maschera
```

```
9d2c eor #$ff   comlemento a uno
9d2e and $0336  e cancella
9d31 tax        quindi stampa
9d32 bne $9d39  il nuovo carattere
9d34 lda #$20   NB. se si 'spegne' un
9d36 jmp $9d3c  carattere non 'legale'
9d39 lda $9c95,x  questo viene
9d3c sta ($d1),y  cancellato.
9d3e ldx $0337  Ripristina la vecchia
9d41 ldy $0338  posizione del cursore
9d44 clc
9d45 jmp $fff0  ed esce.
```

commodore
COMPUTER
CLUB
La rivista degli utenti di sistemi Commodore
Questo mese:
• Vigilante
• Terrore a Dunwich
• Art chess
• Robin Hood
• The OCP Art Studio
• Ferrari F1
• Cube
• Bad Cat
• King of Chicago
• The Bard's Tale
• Phalanx II
• Battleship
• The big deal
• Crazy Cars
• Emetic skimmer
• Mikes
• Champion ship football
• Firepower
• Hollywood poker
• Deja VV
• Mission Elevator
• Insanity flight
• Into the Eagle's nest
23
SUPERGIOCHI
DEL MESE

Come leggere le recensioni

Ogni mese, su queste pagine, verranno esaminati e testati i videogame più recenti per i computer Commodore 64 ed Amiga.

Ad ogni descrizione verranno associate una immagine, catturata tra le più belle schermate, ed una breve pagella.

Quest'ultima, pur se, inevitabilmente, frutto di impressioni personali di chi esamina il gioco stesso, ha lo scopo di assegnare una valutazione del livello del software, soprattutto tenendo conto di altri game analoghi disponibili sul mercato.

La pagella, comprende cinque voci:

IMPATTO: indica il livello di interesse suscitato dalla presentazione e dal tema del gioco.

SCENARIO: riguarda l'accuratezza con cui è realizzata la grafica e l'efficacia dei disegni degli sprite.

SUONO: valuta gli effetti sonori presenti e le eventuali musiche di sottofondo.

INTERESSE: si riferisce al livello di interesse che il gioco può suscitare in un giocatore abituato ai videogame e, implicitamente, alla sua probabilità di "permanenza" sui vostri monitor.

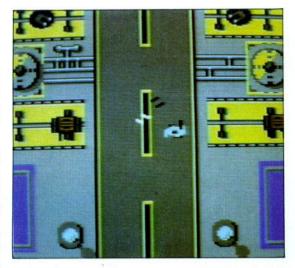
TOTALE: ha lo scopo di sintetizzare i precedenti valori con un voto unico.

VIGILANTE

C/64-128

In questo gioco, ispirato al recente film Robocop, il giocatore, un "Vigilante" un poliziotto bionico computerizzato, deve difendere la metropoli dalla delinquenza, che nell'anno 2011 ha raggiunto livelli catastrofici.

L'organizzazione criminale, conscia della seria minaccia rappresentata dal robot al servizio della legge, tenta in tutti i modi di neutralizzarlo, dandogli la caccia con altri automi e con varie macchine da combattimento.

Il Vigilante è circondato: deve forzare il blocco degli avversari ed inoltrarsi nel territorio urbano fino a raggiungere la base nemica, che si trova in una zona industriale della città; distrutto il cervello dell'organizzazione di macchine criminali, egli riuscirà a riportare la giustizia sul suolo metropolitano.

A mano a mano che il gioco procede, si incontrano aversari sempre più pericolosi.

Solo il coraggio e la bravura del giocatore consentiranno di completare la rischiosa missione!

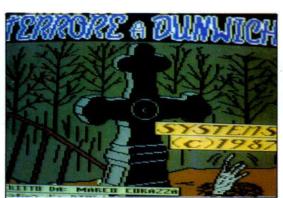
Questo gioco, facente parte della collana "Software made in Italy", è presente assieme ad altri sulla cassetta "Software Club" n.18.

TERRORE A DUNWICH C/64-128

Sullo stesso stile di Zagor, anche "Terrore a Dunwich" propone una bella avventura, questa volta con sfondi tra l'orrore e il macabro...

Come sempre, buona parte dell'avventura è illustrata, ed arricchita da numerosi effetti sonori.

Tu, protagonista, ti troverai immerso in una serie di eventi che difficilmente riuscirai a dominare; dovrai ricorrere al buon senso e alla fede nella logica: a Dunwich, infatti, stanno accadendo cose che di logico hanno ben poco...

Terrore a Dunwich sarà presto in edicola con il disco "Commodore 64 Club" n.5.

ART CHESS Amiga

Una versione di scacchi tridimensionale curata soprattutto nei preziosismi grafici superflui. E' possibile, ad esempio, visionare di continuo, mediante animazione di frecce, la sequenza di mosse "pensate" dal computer, oppure trasformare i quadrati della scacchiera in figure irregolari.

Sicuramente inferiore al buon vecchio Chessmaster della E-CA, per livelli, qualità, numero di opzioni e bravura.

IMPATTO:	7
SCENARIO:	6
SUONO:	5
INTERESSE:	5
TOTALE:	5

ROBIN HOOD
C/64-128

Anche Robin Hood fa parte di "Software Club" n.18 e vede il giocatore nei panni del famoso eroe che ruba ai ricchi per dare ai poveri.

Il coraggio e l'astuzia di Robin risulteranno indispensabili per completare la missione, rischiosa in più parti del gioco.

Attraverso i possedimenti di Sir John, il nostro eroe dovrà combattere e superare un gran numero di ostacoli fino a raggiungere il castello del malvagio signore.

Una particolarità del gioco è data dal fatto che, utilizzando due Joy, è possibile far partecipe dell'avventura anche il fraterno amico Little John.

E... se non avete molta dimestichezza con arco e frecce forza, tutti ad allenarsi!

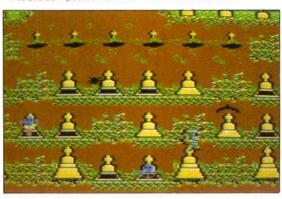
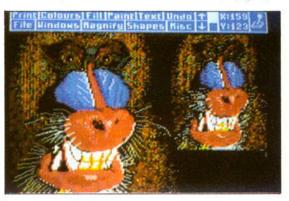
THE ADVANCED OCP ART STUDIO
C/64-128

Anche se non si tratta di un videogame, ma di una utility, questo programma merita senz'altro un posto di rilievo in questa rubrica.

Si tratta senz'altro del miglior programma grafico oggi in commercio per il C/64, in quanto offre una serie di opportunità che difficilmente si trovano in altri tool grafici.

E' infatti predisposto per funzionare sì in Hi-Res che in Multicolor e prevede la possibilità di gestione tramite Mouse.

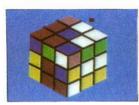
L'interfaccia utente, manco a dirlo, è pefettamente comprensibile e tramite le finestre risulta assai piacevole.

Per ovvi motivi non è possibile inserire la solita pagella.

FERRARI F1
Amiga

Una bella simulazione di gara in formula uno, completa non solo della classica fase di guida in tempo reale (tipo il glorioso POLE POSITION Atari), ma anche delle fasi "manageriali" dove si deve controllare e mettere a punto la macchina.

Il controllo è gestito interamente tramite topo (i tasti fungono da freno e acceleratore) mentre la grafica è coloratissima e molto rifinita.

IMPATTO:	9
SCENARIO:	8
SUONO:	7
INTERESSE:	8
TOTALE:	8

CUBE
C/64-128

Finalmente, per tutti gli appassionati, il famoso Cubo di Rubik in versione informatica.

Fedele in tutto e per tutto, metterà a dura prova la vostra pazienza e la vostra tenacia.

Tutti gli schemi di soluzione adottati per il "vero" cubo funzionano perfettamente nella simulazione proposta.

Tramite la semplice pressione di due tasti potrete giocare a piacimento con il più originale e diffuso rompicapo.

CUBE, come Vigilante e Robin Hood, è presente sulla cassetta "Software Club" n.18, distribuita in tutte le edicole.

BAD CAT
Amiga

Una grafica variegata ed originale per un simpatico micione che deve compiere un percorso ad ostacoli (vedi PITSTOP).

Ogni gioco occupa un solo schermo, alla fine del quale viene caricato il successivo da uno dei due dischetti. L'idea è abbastanza scontata, ma la collezione di giochini, risolti solo grazie a tempismo e rapidità di polso, si rivela piuttosto accattivante.

IMPATTO:	7
SCENARIO:	8
SUONO:	6
INTERESSE:	8
TOTALE:	7

KING OF CHICAGO
Amiga

Un nuovo prodotto della Cinemaware, famosa mamma di SDI, Defender of the Crown e Sinbad. Il gioco è una specie di film ambientato nell'America di Al Capone. Il giocatore deve diventare il re di Chicago, trattando con i malfattori e manovrando operazioni delittuose (alla faccia del gioco educativo!).

La grafica è di un realismo spaventoso, migliore dei cartoni animati giapponesi.

IMPATTO:	9
SCENARIO:	10
SUONO:	6
INTERESSE:	5
TOTALE:	7

THE BARD'S TALE
Amiga

Un gioco di fantasia a ruoli, dove si creano e controllano sei personaggi per vagare nella città di Skara Brae cercando di distruggere il cattivo mago Mangar in Nero. Come in ogni avventura, vi sono cattivi da menare, oggetti da collezionare e luoghi da investigare.

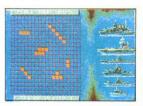
IMPATTO:	6
SCENARIO:	7
SUONO:	4
INTERESSE:	6
TOTALE:	6

PHALANX II
Amiga

Un tipico gioco tipo "spara-e-fuggi" spaziale. Si manovra col joystick un'astronave e si deve sparare a tutto ciò che si muove. Gli oggetti nemici sono di parecchi tipi diversi, con varie sagome di manovra e tutti molto veloci. La scena di sfondo mostra un'enorme struttura spaziale (vi ricorda niente?).

Consigliato solo ai patiti del genere.

IMPATTO:	5
SCENARIO:	6
SUONO:	6
INTERESSE:	5
TOTALE:	5

BATTLESHIP
Amiga

Il classicissimo gioco della battaglia navale, giocabile da una, due o più persone contemporaneamente. La fase di sparo è una realistica e graziosa simulazione arcade di cannoneggiamenti. Prodotto dalla rinomata software house Elite, è certamente un programma originale, per le rifiniture che presenta, e sicuramente godibile da molti.

IMPATTO:	8
SCENARIO:	8
SUONO:	7
INTERESSE:	6
TOTALE:	7

THE BIG DEAL
Amiga

Manovrare un computer che deve provvedere alla gestione di un locale per pranzi "fast-food" (così detto perchè più che cibo è veloce...) è una idea piuttosto originale. Se poi il tutto è supportato da una grafica ineccepibile, a finestre ed animatissima, da effetti sonori notevolmente realistici, il tutto può certamente interessare molti videogiocatori.

IMPATTO:	8
SCENARIO:	8
SUONO:	8
INTERESSE:	7
TOTALE:	8

CRAZY CARS
Amiga

La solita gara automobilistica, con visione non dal cruscotto ma della nostra macchina (si inizia con una mercedes 560). Non si rovina mai la macchina nei fuoristrada, mentre molto spesso anche lasciando il joystick a riposo la macchina prosegue la corsa come se nulla fosse. Le altre macchine che si incontrano sembrano sempre ferme. La grafica di scena è bene gestita e molto colorata. I suoni sono decenti, ma il realismo è molto scadente.

IMPATTO:	6
SCENARIO:	5
SUONO:	6
INTERESSE:	5
TOTALE:	5

EMETIC SKIMMER
Amiga

Si tratta di un originale videogioco spaziale su due dischetti, dove si manovra una specie di bolla spaziale che spara per demolire le basi nemiche intorno, evitando comete e meteore. Il movimento provoca uno scorrimento continuo dello schermo molto fluido, mentre la complessiva scena di gioco è di dimensioni enormi e sconosciute. Non vi sono livelli di difficoltà ed è piuttosto difficile iniziare.

IMPATTO:	7
SCENARIO:	7
SUONO:	6
INTERESSE:	7
TOTALE:	7

MIKES
Amiga

Si controlla un simpatico dragone(!) in un classico gioco "a piattaforme" tipo il glorioso "Miner 2049er". Bisogna cioè aggirarsi su scenografie dalla grafica preziosa raccogliendo oggetti, saltando su ascensori ed evitando i fantasmi. Vi sono almeno una trentina di diversi schermi, difficilissimi. Si può giocare anche in più persone e sul disco possono essere conservati i records di ognuno.

IMPATTO:	8
SCENARIO:	9
SUONO:	8
INTERESSE:	7
TOTALE:	8

CHAMPIONSHIP BASEBALL Amiga

Una gradevole versione del gioco del baseball, con grafi tridimensionali, in tempo reale, molto realistica. Difetto del gioco è la scarsa interazione del giocatore, che si limita a pigiare il tasto del joystick mentre tutto pare muoversi da solo. Graficamente è forse superiore a EARL WEAVER della ECA, ma certamente è molto meno rifinito.

IMPATTO:	7
SCENARIO:	6
SUONO:	7
INTERESSE:	5
TOTALE:	6

FIREPOWER
Amiga

Si può giocare in una o due persone e si manovra, col joystick, un solo carrarmato (oppure due contemporaneamente, su due fette di schermo, se si è in due), per raggiungere la bandiera dell'avversario e riportarla alla base. Si guadagnano punti uccidendo nemici, abbattendo elicotteri e portando i feriti nelle sezioni della croce rossa. Bisogna ovviamente evitare le mine ed il fuoco delle torrette avversarie.

La grafica presenta una visione dall'alto ed è molto graziosa, colorata e ben animata, in grado di entusiasmare grandi e piccoli (videomani). La varietà delle operazioni, la trama avventurosa e la simpatia della grafica lo rendono appetibile per tutti.

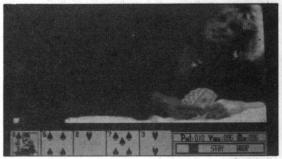
IMPATTO:	9
SCENARIO:	9
SUONO:	8
INTERESSE:	8
TOTALE:	8

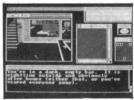
HOLLYWOOD POKER
Amiga

Si tratta del solito STRIP POKER, in cui le nostre avversarie si tolgono capi di vestiario per pagare i debiti di gioco accumulati con noi.

Questo genere di gioco ha interessato molta gente nelle versioni graficamente povere per C/64 ed Apple; figurarsi ora che le quattro "sventole", di chiaro fascino teutonico, spogliabili in quattro fasi, sono riprodotte in buona grafica HAM (4096 colori) con grafica 320 x 100.

L'algoritmo di gioco, scritto in C, non è dei più entusiasmanti, mentre il controllo col joystick avrebbe potuto essere migliore, così come le sintesi vocali, gli effetti sonori ed i disegni delle carte.

IMPATTO:	6
SCENARIO:	5
SUONO:	5
INTERESSE:	6
TOTALE:	5

DEJA VU
Amiga

Una avventura dove siamo nei panni di un un poveraccio che ha perduto la memoria e deve riscoprire chi è. La grafica è continua, non animata, ma con suoni. Il controllo avviene tramite mouse e opzioni predefinite: basta indicare sul video ciò cui siamo interessati ad agire e poi scegliere una parola (examine, open, hit, consume...).

IMPATTO:	7
SCENARIO:	7
SUONO:	6
INTERESSE:	7
TOTALE:	7

MISSION ELEVATOR
Amiga

Si tratta di un vecchio gioco "da bar" implemetato tale e quale (forse più bello) su Amiga. Si controlla una spia che deve indagare in un palazzo a molti piani ed ascensori, sparando a tutte le spie avversarie ed evitando i loro colpi.

Il gioco è un tipico "arcade" che deve il proprio fascino alla semplicità o banalità, secondo i punti di vista.

IMPATTO:	5
SCENARIO:	6
SUONO:	6
INTERESSE:	6
TOTALE:	6

INSANITY FIGHT
Amiga

Un gioco tipo spaziale tuttogrilletto tipo "Plutos" di velocità impressionante (vedere per credere) anche rapportato ai videogiochi "professionali" da bar. La scena prevede enormi piattaforme da sorvolare, astronavi aliene ed ostacoli di ogni tipo. La grafica è ottima e nitidissima (sul monitor...), gli effetti sonori validi, l'idea banale, la protezione anticopia dell'originale eccellente.

IMPATTO:	7
SCENARIO:	8
SUONO:	8
INTERESSE:	8
TOTALE:	7

INTO THE EAGLE'S NEST
Amiga

Si controlla un soldatino inglese all'interno di un castello a sei piani occupato dai nazisti durante la seconda guerra mondiale. Bisogna liberare i nostri amici prigionieri, ammazzare più nemici possibile, collezionare tesori e fare saltare il castello. Il gioco prevede più livelli e molte fasi. La scena di gioco complessiva è enorme.

IMPATTO:	8
SCENARIO:	8
SUONO:	5
INTERESSE:	7
TOTALE:	7

UNO SCHERMO DI CARTA PER IL COMMODORE 64

Routine di hard copy, in modo testo, non ve ne sono mai abbastanza

di **Giancarlo Mariani**

Per i patiti della carta stampata, ecco una nuova routine di hard-copy dello schermo; questa volta non si tratta della "solita" copia dello schermo grafico: la routine proposta permette la stampa dello schermo in modo testo, ossia lettere, numeri, simboli grafici, e via dicendo.

La particolarità del listato risiede nel fatto che è, in un certo modo, "intelligente", ossia riconosce se lo schermo si trova in modo maiuscolo-semigrafico oppure maiuscolo-minuscolo, e stampa di conseguenza.

Hard copy intelligente (27535-27706)

Come tutti sapranno, lo schermo nel C/64 è posizionato a partire dalla locazione $0400 (dec. 1024) fino a $07E7 (2023). In questi 1000 bytes sono contenuti i codici degli altrettanti caratteri presenti sullo schermo, riportati nella tabella del manuale del computer; tali valori sono compresi tra 0 e 127 (caratteri normali) e tra 128 e 255 (stessi caratteri, ma in campo inverso).

Per sincerarvene, battete in modo diretto la linea basic:

For k=0 to 255: poke 1024+k,k:
poke 55296+k,1: next

Vedrete apparire l'intero set di caratteri visualizzabile.

I codici di schermo, purtroppo, sono diversi dai codici Ascii; dal momento che la stampante accetta solo codici Ascii, è necessario convertire il carattere, prima di stamparlo.

Sempre confrontando le tabelle (codici di schermo e codici Ascii), si ricavano semplici regole di conversione seguite, appunto, nella routine qui pubblicata.

Chiamando "C" il codice del carattere, le regole da rispettare sono le seguenti:

1) Se C risulta maggiore di 127, si sottrae 128 a C (C=C-128) e si predispone la stampa in campo inverso.
2) Se C è minore di 32 oppure maggiore di 95, si aggiunge 64 a C (C=C+64) e si salta al punto 4.
3) Se C è compreso tra 64 e 95, allora si aggiunge 32 a C (C=C+32).
4) Se C=34 (cioè carattere di virgolette) si sostituisce con C=39 (accento) per evitare malfunzionamenti di vario tipo.
5) A questo punto, finalmente, il carattere viene stampato.

I punti da 1 a 3 effettuano la conversione appena accennata, mentre il punto 4 sostituisce il carattere apici (") con quello dell'accento (').

Ciò è necessario perchè inviando gli apici alla stampante, questa si predispone in un certo modo nel quale vengono stampati, e non eseguiti, tutti i caratteri di controllo inviati alla stampante stessa (RVS ON, OFF, ecc.) e quindi non si avrebbe una copia fedele dello schermo.

Per ovviare al problema vi sono due soluzioni; la prima consiste nella sostituzione degli apici con l'accento, che non modifica, sostanzialmente, la copia dello schermo; la seconda è piu complessa, perchè pur stampando correttamente gli apici, avrebbe allungato notevolmente la routine, rendendone noiosa la battitura e la comprensione.

Ovviamente è stata scelta la prima soluzione.

Una volta stampato il carattere, si incrementa il contatore, in modo che ogni 40 caratteri venga inviato un return e la stampante si posizioni all'inizio di una nuova linea. Per questa operazione, sulla 801/803 esistono due modi per ottenere altrettanti risultati:

1) Inviare direttamente il CHR$(13) (cioè un return).
2) Inviare, prima del return, un CHR$(8) e, subito dopo il return, un CHR$(15).

La seconda procedura fa in modo che non venga lasciata alcuna spaziatura tra una riga e l'altra. Ciò risulta utilissimo per stampare disegni eseguiti con caratteri semigrafici, mentre riduce la leggibilità di righe di testo consecutive.

La routine di queste pagine, comunque, prevede entrambe le opzioni, selezionabili grazie ad un parametro.

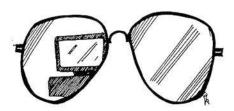
La routine, come detto all'inizio, è intelligente, ossia è in grado di riconoscere lo "stato" (maiuscolo o minuscolo) dello schermo, esaminando la locazione $D018 (53272).

Il bit 2 di questa locazione vale 0 se lo schermo si trova in modo maiuscolo, mentre vale 1 se è in modo minuscolo.

Per cambiare il modo di stampa (ci riferiamo, sempre, alla 801/803 e compatibili), il modo più semplice è aprire il canale con indirizzo secondario uguale a 7:

OPEN 1,4: Predispone la stampa in maiuscolo.
OPEN 1,4,7: Predispone la stampa in minuscolo.

La routine esamina la locazione $D018 e decide automaticamente il valore dell'indirizzo secondario.

A questo punto non resta che eseguire le operazioni descritte in precedenza fino a quando viene raggiunta l'ultima locazione di schermo, ossia la 2023. In seguito viene chiuso il canale e si torna al Basic.

La sintassi

La sintassi della routine presentata è semplicissima:

SYS 27535,V

in cui 27535 è l'indirizzo di partenza consigliato (per rispettare lo standard dell'enciclopedia in l.m.), che può, tuttavia, essere una qualsiasi locazione RAM, grazie alla rilocabilità delle routine, mentre V è il parametro relativo alla spaziatura tra le linee: con V=0 le righe saranno spaziate, mentre con altri valori non vi sarà alcuno spazio tra righe successive.

Il disassemblato commentato, come al solito, descrive in dettaglio il funzionamento della routine.

```
1000 PRINTCHR$(147):PRINT"HARDCO
     PY MODO TESTO"
1010 PRINT:PRINT"SYS XXXX,V"
1020 PRINT:PRINT"V=0: SPAZIATURA
      TRA LE LINEE"
1030 PRINT:PRINT"V<>0: LINEE ATT
     ACCATE"
1040 RETURN
1050 :
1060 REM ** DATI **
1070 :
1080 DATA 032,253,174,032,158,18
     3,134,252,169,000,133,002,1
     33,250,169,004,133
1090 DATA 251,173,024,208,041,00
     2,240,004,169,007,133,002,1
     69,016,162,004,164
1100 DATA 002,032,186,255,169,00
     0,032,189,255,032,192,255,1
     62,016,032,201,255
1110 DATA 160,000,177,250,201,12
     8,144,010,056,233,128,072,1
     69,018,032,210,255
1120 DATA 104,201,096,144,005,02
     4,105,064,208,018,201,032,1
     76,004,105,064,208
1130 DATA 010,201,064,144,006,20
     1,096,176,002,105,032,201,0
     34,208,002,169,039
1140 DATA 032,210,255,169,146,03
     2,210,255,230,254,165,254,2
     01,040,208,023,169
1150 DATA 000,133,254,165,252,24
     0,005,169,008,032,210,255,1
     69,013,032,210,255
1160 DATA 169,015,032,210,255,23
     0,250,208,002,230,251,165,2
     51,201,007,208,154
1170 DATA 165,250,201,232,208,14
     8,169,013,032,210,255,032,2
     04,255,169,016,076
1180 DATA 195,255,-1,23115
```

```
C000 jsr $aefd      ;Controllo virgola.
C003 jsr $b79e      ;Prende parametro.
C006 stx $fc        ;Lo salva in $FC.
C008 lda #$00       ;Azzera loc. x minusc.
C00a sta $02        ;
C00c sta $fa        ;Inizializza $FA/$FB
C00e lda #$04       ;a $0400 (mem. di
C010 sta $fb        ;schermo).
C012 lda $d018      ;Controlla se schermo
C015 and #$02       ;e' in maius. o minus.
C017 beq $C01d      ;
C019 lda #$07       ;Valore per mettere
C01b sta $02        ;stampante in minusc.
C01d lda #$10       ;Apre il canale
C01f ldx #$04       ;sulla stampante
C021 ldy $02        ;(OPEN 16,4).
C023 jsr $ffba      ;
C026 lda #$00       ;
C028 jsr $ffbd      ;
C02b jsr $ffC0      ;Apre.
C02e ldx #$10       ;Esegue CMD 16.
C030 jsr $ffc9      ;
C033 ldy #$00       ;Carica un carattere
C035 lda ($fa),y    ;da mem. schermo.
C037 cmp #$80       ;E' >=128?
C039 bcc $C045      ;No: Salta a $C045.
C03b sec            ;Si: Sottrai 128
C03c sbc #$80       ;
C03e pha            ;e metti la stampa
C03f lda #$12       ;in campo inverso.
C041 jsr $ffd2      ;
C044 pla            ;
C045 cmp #$60       ;E' >=96?
C047 bcc $C04e      ;No: Salta a $C04E.
C049 clc            ;Si: Aggiungi 64
C04a adc #$40       ;
C04c bne $C060      ;e salta a $C060.
C04e cmp #$20       ;E' <32?
C050 bcs $C056      ;No: Salta a $C056.
C052 adc #$40       ;Si: Aggiungi 64
C054 bne $C060      ;e salta a $C060.
C056 cmp #$40       ;E' >=64?
C058 bcc $C060      ;No: Salta a $C060.
C05a cmp #$60       ;Si: E' <96?
C05c bcs $C060      ;No: Salta a $C060.
C05e adc #$20       ;Si: Aggiungi 32.
C060 cmp #$22       ;Sono apici?
C062 bne $C066      ;No: Salta a $C066.
C064 lda #$27       ;Si:Cambia con accento
C066 jsr $ffd2      ;Stampa il carattere.
C069 lda #$92       ;Stampa RVS OFF.
C06b jsr $ffd2      ;
C06e inc $fe        ;Incrementa cont. car.
C070 lda $fe        ;E' arrivato a 40
C072 cmp #$28       ;(cioe' fine riga)?
C074 bne $C08d      ;No: Salta a $C08D.
C076 lda #$00       ;Si: Azzera contatore
C078 sta $fe        ;(per nuova riga).
C07a lda $fc        ;Spaziatura tra righe?
C07c beq $C083      ;Si: Salta a $C083.
C07e lda #$08       ;No: Manda un chr$(8).
C080 jsr $ffd2      ;
C083 lda #$0d       ;Stampa un return
C085 jsr $ffd2      ;(cioe' nuova linea).
C088 lda #$0f       ;Invia un chr$(15)
C08a jsr $ffd2      ;(caratteri normali).
C08d inc $fa        ;Incrementa puntatore
C08f bne $C093      ;a memoria di
C091 inc $fb        ;schermo.
C093 lda $fb        ;E' arrivato a 2023
C095 cmp #$07       ;(ultimo carattere)?
C097 bne $C033      ;No: ricomincia.
C099 lda $fa        ;
C09b cmp #$e8       ;
C09d bne $C033      ;Idem C.S.
C09f lda #$0d       ;Si: stampa un return.
C0a1 jsr $ffd2      ;
C0a4 jsr $ffcc      ;Resetta canali I/O.
C0a7 lda #$10       ;Chiude il canale 16.
C0a9 jmp $ffc3      ;(CLOSE 16).
```

ESPANSIONE DI MEMORIA PER C/64

Che cosa si può aggiungere ad un C/64 in modo da farlo competere con elaboratori più "grossi" di lui? Ma è chiaro: altri 256 K di memoria RAM

di **Alessandro de Simone**

Quando il C/64 fu progettato, decisero di inserire al suo interno la massima quantità di memoria RAM disponibile per un microprocessore ad 8 bit. A questa decisione giunsero approfittando del fatto che, in quei tempi, era iniziata in modo massiccio la produzione di memorie da 64 K su un solo chip.

Da allora molto tempo è passato e la norma sembra essere dettata dalla capacità di un mega di bit per ciascun circuito integrato.

Nonostante il 6510, il microprocessore installato sul C/64, possa indirizzare solo 64 K di memoria, è possibile, manipolando opportunamente alcuni banchi di memoria "sovrapposti" tra loro, selezionarne uno solo di essi in modo che il 6510 "veda" solo quello ed il suo contenuto.

In pratica, selezionando uno dei banchi di memoria disponibili, è come se si potesse disporre di altrettanti computer C/64, contemporaneamente presenti ed attivi.

Il modulo 1764

La capacità di 256 KRam di cui è dotata l'espansione di memoria 1764, ovviamente, non è utilizzabile direttamente da Basic.

E' bene dire subito, per evitare cocenti delusioni, che non solo il software a corredo è riportato su disco (e chi ha il solo registratore, purtroppo, si attacca al tram), ma la memoria disponibile per il programmatore in Basic,

Corso di Linguaggio Macchina e routine Grafiche per il tuo Commodore 64

Una pubblicazione monografica della Systems Editoriale, curata da Alessandro de Simone, per avvicinarsi al meraviglioso mondo del Linguaggio Macchina e della sua più immediata applicazione: la grafica tridimensionale in alta risoluzione.

Richiedi oggi stesso la super-confezione contenente il fascicolo "Commodore Speciale", un fascicolo omaggio di Commodore Computer Club ed il dischetto che riporta TUTTI i programmi pubblicati nello stesso fascicolo!

N.B.: Il solo fascicolo "Commodore Speciale" è offerto al prezzo di L.6000 oltre a L.3000 per spese di spedizione.

Non è possibile inviare i programmi su nastro-cassetta, ma solo su disco.

Coloro che desiderano procurarsi il package completo (due fascicoli oltre al dischetto) devono utilizzare la scheda pubblicata in fondo alla rivista.

all'accensione della macchina, è ancora di 38911 bytes.

A che serve, dunque, il nuovo accessorio?

Serve, anzitutto, come drive "virtuale" di ben 989 blocchi da 256 byte ciascuno (per un totale, appunto, di 253184 byte) ad altissima velocità di accesso.

Inoltre, per chi ci sa fare, può servire per memorizzare numerose pagine grafiche in alta risoluzione idonee a creare stupefacenti animazioni.

Nel primo caso, ad esempio, si verificano caricamenti e salvataggi talmente veloci che non è possibile cronometrare il tempo necessario all'operazione: il cursore torna infatti a lampeggiare quasi subito dopo la pressione del tasto Return, anche trattando programmi lunghissimi.

Non risulta necessaria, pertanto, alcuna esperienza di programmazione: nelle operazioni di salvataggio e caricamento di programmi con il drive "virtuale" basta sostituire il suffisso ",8" con ",9".

Nel secondo caso, però, (ci riferiamo alla memorizzazione di più immagini in alta risoluzione) occorre smanettare con i banchi e con varie Poke Peek e Sys; nulla di trascendentale, intendiamoci, ma la conoscenza del linguaggio macchina non guasta.

Le prove cui è stato sottoposto il modulo 1764, fornitoci gentilmente dalla Ditta Niwa di Sesto San Giovanni (che ne garantisce la disponibilità ai suoi affezionati clienti) ha funzionato egregiamemente sia con un C/64 nuovo che con un esemplare della vecchia generazione.

Importante è sottolineare che l'accessorio conserva la totale compatibilità con lo Speed Dos e che, con un po' di pazienza (cioè spostando i puntatori di inizio e fine memoria per i programmi in l.m.), è possibile riportare su nastro i vari programmi-utility presenti sul disco allegato: la RamDisk, infatti, funziona perfettamente anche con un registratore a cassette e non si capisce perchè la Commodore non abbia allegato, nella confezione, anche il nastro, in modo da favorire i potenziali clienti.

Le note negative

A onor del vero c'è ben poco da sparlare!

La gestione dei banchi è semplice (per chi se ne intende, lo ripetiamo) e l'incompatibilità con alcuni programmi si risolve non attivando le utility presenti oppure (ma in casi molto, molto rari) rimuovendo l'accessorio prima di accendere il computer.

Nella confezione è presente anche un alimentatore. Questo, più potente di quello di cui sono dotati i C/64, consente di alimentare adeguatamente non solo il computer, ma anche la notevole vastità di Ram aggiunta.

Potevano certo spendere un migliaio di lire in più per dotare l'alimentatore di un interruttore (come sull'Amiga), ma, chissà perchè, non l'hanno ritenuto necessario.

Seccante è l'assenza del comando Verify; imbarazzante (ma è un eufemismo) il caso in cui si tenti di registrare un programma dotato di un nome già esistente nella directory virtuale: il programma non viene registrato, ma nessun messaggio compare per avvertire dell'impossibilità di effettuare l'operazione.

Quasi tutti i comandi del tipo B-R, B-W e simili non sono accettati.

Concludendo

L'espansione di memoria 1764 rappresenta certamente un valido strumento per aumentare le potenzialità del C/64.

Si consiglia a coloro che non intendono rinunciare al piccolo Commodore per nessun motivo al mondo.

Si tenga presente, però, che solo il Geos, almeno finora, può sfruttare l'espansione fino in fondo: tutti gli altri programmi professionali (e non) continueranno ad essere "insensibili" alla maggiore disponibilità di memoria. Ci sarà qualche software house che si darà da fare per proporre nuovi package? Staremo a vedere..

...E CENTO!!!

La routine di queste pagine è un po' particolare, non tanto per la sua utilità, quanto perchè...

20000 Arrotondamento
(Qualsiasi Commodore)

Questa semplice routine, che potrete usare in moltissime occasioni, è stata sviluppata approfittando della richiesta di un nostro lettore,(Renzo Manzatto di Cegga).

Consente di fissare il numero massimo di cifre eventualmente presenti dopo la virgola di un numero decimale.

Benchè l'argomento sia stato affrontato (molto) tempo fa, invitiamo i lettori ad esaminare la routine.

Da notare, anzitutto, che il numero massimo di cifre visualizzabile con un C/64 (C/128, C/16 e Plus/4) è 9; ne consegue che non è possibile pretenderne di più (riga 20000).

Le altre righe Basic si limitano a moltiplicare il numero in oggetto (X8) per il numero "1" seguito da tanti zeri quante sono le cifre desiderate dopo la virgola. Subito dopo viene "estratta" la parte intera (riga 20020) e, in seguito, riconvertita dividendo il numero ottenuto (X7) per la stessa potenza di 10.

Nel caso in cui il numero decimale da trattare risultasse negativo, è necessario elaborarlo in modo diverso (usando come deviatore la variabile X4) per evitare noie in fase di arrotondamento.

La routine, nonostante sia universale, presenta alcuni limiti. Il primo è quello relativo al trattamento di numeri espressi in forma esponenziale (come: 1E22) che dovrebbero esser trattati a parte.

L'altro limite è rappresentato dal fatto che se le ultime cifre sono nulle (oppure se il numero è intero), queste non vengono visualizzate per ovvi motivi. Per esempio, il numero 1,0000, richiesto con tre cifre dopo la virgola, dovrebbe fornire 1,000 mentre la routine risponde, giustamente, soltanto con: 1.

Ai più volenterosi che desiderassero a tutti i costi la visualizzazione degli zeri, daremo solo un suggerimento: dopo aver trasformato il valore elaborato (X6) nella corrispondente stringa (mediante STR$) controllate se è presente il simbolo della virgola (che è un... punto) ed il numero di cifre presenti dopo di essa; quindi aggiungete tanti caratteri zero fino a raggiungere la quantità richiesta.

Pensate che sia facile? mettetevi all'opera e cronometrate il tempo che dedicate allo sviluppo dell'algoritmo, ma mi raccomando: fate in modo che la modifica sia valida per qualsiasi valore (positivo o negativo) e qualunque sia il numero di cifre richieste...

C'era una volta

Con il n.24 di Commodore Computer Club (ottobre 1985) iniziò la pubblicazione della nota "Enciclopedia di routine" in Basic, voluta dagli stessi lettori che desideravano entrare in possesso di elementari procedure per svolgere diversi compiti.

Lo standard adottato, semplice ma efficace, incontrò il favore degli utenti Commodore che, oltre alla semplicità di digitazione, trovarono una miniera di idee da sviluppare per proprio conto.

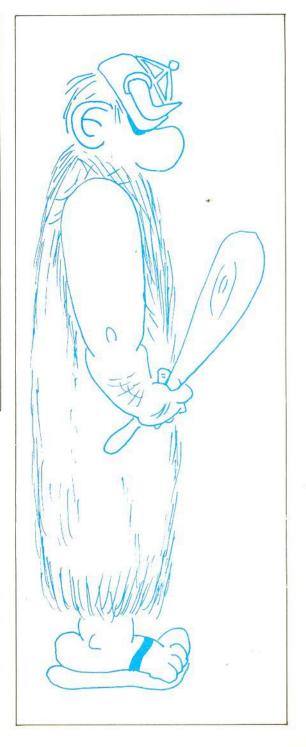
Quasi contemporaneamente alla nascita dell'Enciclopedia, fu soppressa l'altra rubrica, "Una riga", che offriva micro-programmi autosufficienti lunghi... una sola riga Basic!

Come, allora, la proposta di una rubrica più ampia, come l'Enciclopedia, tendeva a sostituire l'inevitabile esaurirsi della "vena" di "Una Riga", così oggi, gira e rigira, dobbiamo riconoscere che (quasi) tutto ciò che poteva esser sviluppato in una sola schermata è stato ormai pubblicato.

La routine di questo numero porta il n.100 ma, in effetti, ne sono state pubblicate in maggior numero perchè alcune sono state riproposte migliorate ed ampliate rispetto a versioni precedenti.

Stiamo forse per dire che con questo numero termina la pubblicazione dell'Enciclopedia? Bè, in parte sì e in parte no.

E con tale frase sibillina vogliamo invitare i lettori a seguirci nel prossimo numero...

```
   50 REM DEMO ARROTONDAMENTO
   60 :
   70 X8=1.123456789:FOR I=1 TO 8
   80 X9=I:GOSUB 20000:PRINTX6:NE
      XT
  100 INPUT "N.DECIMALE";X8
  105 INPUT "N.CIFRE";X9
  107 GOSUB 20000:PRINTX0$
  110 PRINT"N. ELABOR.";X6
  120 GOTO 100
 9998 :
 9999 END
20000 X4=0:X0$="":IF X9>9 THEN X0
      $="ERR":RETURN
20010 X5=ABS(X8):IF X5<>X8 THEN X
      4=1
20020 X7=INT(X5*10↑X9)
20030 X6=X7/10↑X9:IF X4=1 THEN X6
      =-X6
20098 RETURN
20099 REM  N.CIFRE DOPO LA VIRGOL
      A
```

QUANTO COSTA IL TUO COMMODORE

Codice	Prodotto	Prezzo (IVA esclusa)
A 500	Personal Computer 16/32 BIT CPU 512K RAM incorpora un floppy disk drive da 3" 1/2 da 880Kb e un mouse	L. 950.000
A 501	Espansione di memoria per Amiga 500 512Kbytes e orologio	L. 212.500
A 520/1	Modulatore per Amiga 500 permette di collegare Amiga 500 ad un qualsiasi televisore b/n e colori	L. 40.000
	Cavo collegamento Amiga 500 con TV prescart	L. 27.000
Nuovo C64	Nuovo personal computer CPU 64K RAM computer ad alta risoluzione grafica, 256 combinazioni di colori, sintetizzatore di suono. Collegabile ad un qualsiasi televisore	L. 375.000
1764	Espansione di memoria per C64 256Kbytes	L. 195.000
C 128D	Personal computer CPU 128Kb RAM CPU 128Kbytes espandibile a 512Kbytes, 48 Kbytes ROM, basic 7.0. Tastiera separata. Alta risoluzione grafica, 16 colori + 8 sprites. 40 80 (RGB) colonne. Programmabile in CP/M 3.0. Funzionante in modo C64. Floppy disk da 340Kb incorporato.	L. 895.000
1700	Espansione di memoria per C128. 128Kb	L. 165.000
1750	Espansione di memoria per C128. 512Kb	L. 235.000
1530	Registratore compatibile C64 - C128 -C128D	L. 55.000
1541 II	Floppy disk drive 5" 1/4. Unità di memoria di massa, drive singolo, capacità 170Kbytes in linea. Compatibile con C64	L. 395.000
1571	Floppy disk drive 5" 1/4. Unità di memoria di massa, drive singolo, capacità. 340Kbytes in linea. 410K sotto CP/M. Compatibile.con C128 - C128D	L 460.000
1581	Floppy disk drive 3" 1/2. Unità di memoria di massa, drive singolo da 3" 1/2, capacità 800Kbytes. Compatibile C64 - C128 -C128D	L. 420.000
A 1010	Floppy disk drive esterno 3" 1/2	L. 495.000
2080	Monitor a colori alta persistenza alta risoluzione, 14", con audio antiriflesso. Collegabile ad Amiga PC, C64, C128, C128D. 640x400 pixel, 4096 colori	L. 570.000
1084	Monitor a colori. Alta risoluzione, 14", con audio, antiriflesso. Collegabile al C64 - C128 - C128D - Amiga - PC. 640x400 pixel, 4096 colori.	L. 570.000
MPS 1200	Stampante. 80 colonne, 120 cps, bidirezionale, carta in modulo singolo e trascinamento modulo continuo per C64 - C128 -C128D	L. 495.000
MPS 1200 P	Come MPS 1200 per A500	
MPS 1500	Stampante a colori. 80 colonne, 130 cps, 4 colori, bidirezionale, carta in modulo singolo e trascinamento modulo continuo.	•
6499	Adattatore telematico omologato. Collegabile al C64, permette il collegamento a Videotel PGE-Banche Dati.	L. 149.000
1851	Mouse. Per C64 C128D	L. 99.000
1311	Joystick. Comando per giochi	L. 13.500

Codice	Prodotto	Prezzo (IVA esclusa)
Amiga 2000	Microprocessore Motorola 68000; clock 7,16 MHz; 1MB RAM; 256KB ROM; kickstart in ROM. Uscita PAL; 4 uscite a 4 voci su 2 canali (stereofonia). Amiga DOS; Amiga multitasking. 7 slots di sistema (2 combinati Amiga - Ibm PC AT); 1 slot video; 1 accesso diretto CPU 86 pin. Compatibilità MS-DOS interna con schede Janus (XT/AT CARD). Possibilità di un secondo drive interno da 3" 1/2 e di un drive interno da 5" 1/4. Monitor a colori 1084 tastiera; mouse. Workbench; Extras	L. 2.550.000
A 1010	Floppy disk drive esterno da 3" 1/2	L. 495.000
A 2080	Monitor a colori ad alta risoluzione, a alta persistenza. 14" con audio; 640x400 pixel; 4096 colori	L. 570.000
A 1084	Monitor a colori ad alta risoluzione. 14" con audio; antiriflesso; 640x400 pixel; 4096 colori	L. 570.000
A 2010	Floppy disk drive interno da 3" 1/2	L. 280.000
A 2092	Hard disk da 20MB settorizzabile. PC/Amiga + scheda controller hard disk	L. 1.085.000
PC 60/40	Microprocessore 80386 a Clock 8/16 MHz memoria RAM 25/2Kbyte. Hard disk da 40Mb, un floppy disk da 1,2Mb. Sistema operativo MS-DOS 3.2. Scheda EGA di serie. Monitor ADI monocromatico a fosfori verdi da 14". Porta seriale RS232C e parallela Centronics. In opzione: floppy disk drive da 3" 1/2 e 1,44Mb; coprocessore matematico 80387.	
PC 60/80	Stessa configurazione del PC 60/40 ma con un hard disk da 80Mb e, in più, MS-DOS Windows 386	
CP 80387	Coprocessore matematico 80387, 16MHz.	
	Floppy disk drive aggiuntivo, 3" 1/2, 1,44Mb.	
PCI	Microprocessore 8088 (coprocessore 8087 opzionale) frequenza clock 4,77MHz. RAM 512Kb. Espandibile a 640Kb; 1 floppy da 360Kb. Monitor a fosfori verdi 12".	
PC-10-II	Microprocessore Intel 8086 a 16 bit; memoria RAM da 640Kb; memoria ROM (Bios) da 8Kb autoconfigurabile. Sistema operativo MS-DOS 3.20. Scheda video monocromatico/colore AGA di serie. Due floppy disk drive da 360Kb. Monitor monocromatico a fosfori verdi da 12". Porta seriale RS232C e parallela Centronics.	L. 1.990.000
PC 20-II	Microprocessore Intel 8088 a 16 bit; memoria RAM da 640Kb; memoria ROM (BIOS) da 8Kb autoconfigurabile. Sistema operativo MS-DOS 3.20. Scheda video monocromatico/colore AGA di serie. Un hard disk da 20Mb e un floppy disk drive da 360Kb. Monitor monocromatico a fosfori verdi da 12". Porta seriale RS232C parallela Centronics.	L. 2.990.000
PC 40/40	Microprocessore 80286 a 16 bit; clock 10/6 MHz; memoria RAM da 1Mb. Un hard disk da 40Mb; un floppy disk drive da 1,2Mb. Sistema operativo MS-DOS 3.2. Scheda video monocromatico/colore 132 colonne AGA di serie. Monitor monocromatico a fosfori verdi da 14". Porta seriale RS232 e parallela Centronics. GW Basic	L. 3.990.000
CP 80287	Coprocessore matematico 80287, 10MHz	L. 690.000
MPS 1200P	Stampante a 9 aghi 120 cps bidirezionale 80 colonne. Interfaccia Commodore per PC 10/III PC 40 PC/60/80	L. 550.000
MPS 2010	Monitor a colori ad alta risoluzione 13" con audio	L. 650.000
A 2080+ A 2020	Scheda Janus XT compatibile + drive Interno da 5" 1/4	L. 1.210.000
A 2286 + A 2020	Scheda Janus II AT compatibile + drive Interno da 5" 1/4	*
A 2995	Scheda 68020 con processore a 32 bit; 14,28 MHZ	*
A 2997	Interfaccia Genlock multistandard professional	L. 3.490.000
A 2998	Interfaccia Genlock VHS/BETA	L. 1.290.000
A2999	Framegrabber	*
	Amiga DOS user manual	L. 26.000
	Textcraft	L. 40.000
	Graphicraft	L. 40.000
	Mind Walker	L. 40.000
	Pascal	L. 90.000
	Lattice C"	L. 90.000
	Lisp	L. 90.000
	De Luxe Paint	L. 130.000
	De Luxe Print	L. 130.000
	De Luxe Video	L. 130.000
	System five Unix	L. 1.290.000
	Superbase in italiano	L. 130.000
	Lo Logistix Amiga	L. 130.000
	Logistix Versione PC	L. 740.000
	Volkswriter 3 in italiano	L. 745.000
	Page setter (desk top publishing) in italiano	L. 180.000

• Prezzo da definire

I primi 100 negozi selezionati dalla "Commodore" per la vendita dei suoi prodotti

Al Risparmio	*V.le Monza 204*	20128 Milano	Tel. 02-2573440
Braha A. s.d.f.	*Via Pier Capponi 5*	20145 Milano	Tel. 02-437468
E.D.S.	*C.so P.ta Ticinese 4*	20123 Milano	Tel. 02-8322045
Faref	*Via A. Volta 21*	20121 Milano	Tel. 02-650042
Giglioni	*V.le Luigi Sturzo 45*	20154 Milano	Tel. 02-654906
Logitek	*Via Golgi 60*	20133 Milano	Tel. 02-538931
Marcucci	*Via F.lli Bronzetti 37*	20129 Milano	Tel. 02-7386051
Melchioni	*Via P. Colletta 37*	20135 Milano	Tel. 02-57941
Messag. Musicali	*Gal. del Corso*	20100 Milano	Tel. 02-50841
Newel	*Via Mac Mahon 75*	20155 Milano	Tel. 02-323492
Rivola	*Via Vitruvio 43*	20124 Milano	Tel. 02-6694160
F.lli Galimberti	*v. Naz. Giovi 28/36*	20030 Barlassina (Mi)	Tel. 0362-560625
P. Giorgio Ostellari	*Via Molino Arese 65*	20031 Cesano M. (Mi)	Tel. 0362-504392
P. Giorgio Ostellari	*Via Milano 300*	20233 Desio (Mi)	Tel. 0362-621042
GBC Italiana	*V.le Matteotti 66*	20092 Cinisello B. (Mi)	Tel. 02-6181801/ 6189391
Casa Della Musica	*Via Indipendenza 21*	20093 Cologno M. (Mi)	Tel. 02-2542117
Penati	*Via Verdi 28/30*	20011 Corbetta	Tel. 02-9779401
EPM System	*V.le Italia 12*	20094 Corsico (Mi)	Tel. 02-4407979
Cen. Comp. Pandolfi	*Via Corridoni 18*	20025 Legnano (Mi)	Tel. 0331-546426
Computeam	*Via Vecellio 41*	20035 Lissone (Mi)	Tel. 039-481010
Futura	*Via Solferino 31*	20075 Lodi (Mi)	Tel. 0371-54457
L'Amico del comp.	*V.le Lombardia 17*	20077 Melegnano (Mi)	Tel. 02-9838341
Bit 84	*Via Italia 4*	20052 Monza (Mi)	Tel. 039-320813
I.C.O.	*Via dei Tigli 14*	20090 Opera (Mi)	Tel. 02-5242146
Comif	*Via Autolinee 10*	24100 Bergamo	Tel. 035-218553
Cordani	*Via dei Caniana 8*	24100 Bergamo	Tel. 035-258184
D.R.B.	*Via B. Palazzo 65*	24100 Bergamo	Tel. 035-237292
New Systems	*Via Paglia 36*	24100 Bergamo	Tel. 035-248109
Comp. Team hi-tec	*Via Verdi 1/B*	24030 Carvico (Bg)	Tel. 035-790244
Ott. Opt. Rovetta	*P.zza Garibaldi 6*	24065 Lovere (Bg)	Tel. 035-960705
A.I.S. International	*Via San Carlo 25*	24016 San Pellegrino Terme (Bg)	Tel. 0345-22662
Sisthema	*Via Roma 45*	24067 Sarnico (Bg)	Tel. 035-910750
Computer Center	*Via Cipro 62*	25125 Brescia	Tel. 030-223230
Informatica 2000	*Via Stazione 16/B*	25100 Brescia	Tel. 030-54015
Vigasio Mario	*P.zza Zanardelli 3*	25100 Brescia	Tel. 030-59330/ 295858
Mister Bit	*Via Mazzini 70*	25043 Breno (Bs)	Tel. 0364-22835
Cavalli Pietro	*Via 10 Giornate 14B*	25030 Castrezzato (Bs)	Tel. 030-714013
Megabyte	*P.zza Duomo 17*	25015 Desenzano del Garda (Bs)	Tel. 030-9144880
Info Cam	*Provinciale 38*	25050 Gratacasolo (Bs)	Tel. 0364-89379
Il Computer	*Via Indipendenza 90*	22100 Como	Tel. 031-240959
2M Elettronica	*Via Sacco 3*	22100 Como	Tel. 031-278227
Elettrongros	*Via L. Da Vinci 54*	22062 Barzanò (Co)	Tel. 039-957318
Ega	*Via Mazzini 42*	22065 Cassago Br. (Co)	Tel. 039-956307
Ega	*Via Aldo Moro 17*	22043 Galbiate (Co)	Tel. 0341-522028
Data Found Comp.	*Via A. Volta 4*	22036 Erba (Co)	Tel. 031-645761
Righi Elettronica	*Via G. Leopardi 26*	22077 Olgiate C. (Co)	Tel. 031-946766
Fumagalli	*Via Cairoli 48*	22053 Lecco (Co)	Tel. 0341-863341
Cima Eelettronica	*Via L. Da Vinci 7*	22053 Lecco (Co)	Tel. 0341-371106
Mondo Computer	*Via Giuseppina 11/B*	26100 Cremona	Tel. 0372-882079
Prisma	*Via B. Da Dovara 8*	26100 Cremona	Tel. 0372-437900
Elcom/GBC	*Via IV Novem. 56/58*	26013 Crema (Cr)	Tel. 0373-83393
Euroelettronica	*Via XX Settembre*	26013 Crema (Cr)	Tel. 0373-86966
Computer	*Galleria Fermi 7*	46100 Mantova	Tel. 0376-325616
32 Bit (Comp. St.)	*Via C. Battisti 14*	46100 Mantova	Tel. 0376-326770
Log. Inf. Comp.	*V.le M. Grappa 32*	27029 Vigevano (Pv)	Tel. 0381-81883
M. Visentin	*C.so V. Emanuele 76*	27029 Vigevano	Tel. 0381-83833
Computer Line	*Via G. Carducci 4*	29100 Piacenza	Tel. 0523-30691
Delta Computer	*Via Mar. Resisten.15/4*	29100 Piacenza	Tel. 0523-753318
Sover	*Via IV Novembre 60*	29100 Piacenza	Tel. 0523-34388
Fotonova		23010 S. Pietro di Berbenno (So)	Tel. 0342-492319
Il Centro Elettr.	*Via Morazzone 2*	21100 Varese	Tel. 0332-231006
Busto Bit	*Via Gavinana 17*	21052 Busto A. (Va)	Tel. 0331-625034
Crespi G. & C.	*V.le Lombardia 59*	21053 Castellanza (Va)	Tel. 0331-503023
Computer Shop	*Via A. Da Brescia 2*	21013 Gallarate (Va)	Tel. 0331-798612
J.A.C. Nuo. Tec.	*Via Matteotti 38*	21018 Sesto C. (Va)	Tel. 0331-923134
Bit Micro	*Via Mazzini 102*	15100 Alessandria	Tel 0131-443252
West Records	*C.so Roma 85*	15100 Alessandria	Tel. 0131-441090
S.G.E. Elettronica	*Via Bandello 19*	15057 Tortona (Al)	Tel. 0131-867709
Rossi Computers	*C.so Nizza 42*	12100 Cuneo	Tel. 0171-2339
Punto Bit	*C.so Langhe 26/C*	12051 Alba (Cn)	Tel. 0173-49833
Curetti Augusto	*C.so Italia 3*	12054 Mondovì (Cn)	Tel. 0174-42014
Ditta Elettrogamma	*c.so Risorgim.to 20*	28100 Novara	Tel. 0321-176358
Elcom	*C.so Mazzini 11*	28100 Novara	Tel. 0321-391293
Programma 3	*V.le Buonarroti 8*	28100 Novara	Tel. 0321-36367/ 399903
Punto Video	*c.so Risorgim.to 391*	28100 Novara	Tel. 0321-477367
Computer	*Via Monte Zeda 4*	28041 Arona (No)	
All Computer	*C.so Garibaldi 106*	28021 Borgomanero (No)	Tel. 0322-844142
Micrologic	*Via Giovanni XIII 2*	28037 Domodossola (No)	
Elliott Comp. Shop	*Via Don Minzoni 32*	28044 Intra (No)	Tel. 0323-43517
ABA Elettronica	*Via C. Fossati 5/P*	10100 Torino	Tel. 011-302065
Alex Comp.&giochi	*C.so Francia 333/4*	10142 Torino	Tel. 011-7730184
C.D.M. Elettronica	*Via Marochetti 17*	10126 Torino	Tel. 011-636345/ 634900
Computing New	*Via Marco Polo 40/E*	10129 Torino	Tel. 011-501512
De Bug	*c.so V. Eman.le II 22*	10100 Torino	Tel. 011-832986
Desme Universal	*Via San Secondo 95*	10100 Torino	Tel. 011-592551
F.D.S.	*Via Borgaro 86/D*	10100 Torino	Tel. 011-2168900
MT Informatica	*C.so G. Cesare 58*	10100 Torino	Tel. 011-850955/ 238803
New Bus. Comp.	*Via Nizza 45*	10100 Torino	Tel. 011-214235
Radio Tv Mirafiori	*C.so Un. Sov.ca 381*	10135 Torino	Tel. 011-616190/ 6197189
SMT Elettronica	*Via Bibiana 83/B*	10147 Torino	Tel. 011-218243
Paul e Chico V.	*Via V. Emanuele 52*	10023 Chieri (To)	Tel. 011-9470295
Bit Informatica	*Via V. Emanuele 154*	10073 Ciriè (To)	Tel. 011-9205455
Eurex	*C.so Indipendenza 5*	10086 Rivarolo C.se (To)	Tel. 0124-27984
Ditta Elettrogamma	*C.so Bormida*	13100 Vercelli	Tel. 0161-53689
Elettronica	*Strada Torino 15*	13100 Vercelli	Tel. 0161-393163
C.S.I. Teorema	*Via Losana 9*	13051 Biella (Vc)	Tel. 015-28622
St. Fotogr. Imarisio	*P.zza M. Libertà 7*	13039 Trino (Vc)	Tel. 0161-82081
F.lli Gatti	*Via Festaz 75*	11100 Aosta	Tel. 0165-35659

A chi chiedere aiuto se il tuo Commodore ha problemi

ABRUZZO

Audio Computer
Via Umbria, 7
65100 Pescara
Tel. 085/29.33.75

CALABRIA

Service Center
Via Parisio, 25
87100 Cosenza
Tel. 0984/75.74.1

CAMPANIA

Electrical Engineer
Via Supportico Lopez, 5/A
80137 Napoli
Tel. 081/44.44.44 - 29.34.08

Marvin Service
Via Nazionale delle Puglie, 344
80013 Casalnuovo (Na)
Tel. 081/84.24.22.4

EMILIA ROMAGNA

Centro Riparatori
Via Lenin, 48/I
41012 Carpi
Tel. 059/64.07.70

Centro Riparatori
Via Galvani, 4
41100 Modena
Tel. 059/21.66.02

Dr. Sax
Via D. Creti, 77/C
40128 Bologna
Tel. 051/35.25.39

Maser
Via Collegio di Spagna, 10
40064 Ozzano Emilia (Bo)
Tel. 051/79.84.48

Centro Raccolta
Via Di Corticella, 177
40128 Bologna

S.I.R.A. srl
Via Aniene, 43/45
48100 Ravenna
Tel. 0544/64.22.3

FRIULI VENEZIA GIULIA

Audio Video Service
Via Gemelli, 9
33170 Pordenone
Tel. 0434/57.11.04

ET Italia
Via Tavagnacco, 89
33100 Udine
Tel. 0432/48.13.39

Hitech
Via Nordio, 9
34100 Trieste
Tel. 040/74.11.89

LAZIO

Computer Service Italia
Via Sebino, 49
00199 Roma
Tel. 06/85.03.86

Tecnicomp
Via dei Georgofili, 65
00159 Roma EUR
Tel. 06/51.33.73.9

Elettrotel srl
Via Verolengo, 20
00167 Roma
Tel. 06/62.31.06.8

LIGURIA

Siragusa
Via Milano, 41
16126 Genova
Tel. 010/26.16.55

Teleradio
Via XXV Aprile, 70
18100 Imperia
Tel. 0183/21.96.2

LOMBARDIA

Abiservice srl
Via Ponale, 48
20100 Milano
Tel. 02/64.37.49.6

Catme
Via Severoli, 9
20147 Milano
Tel. 02/41.52.96.2

Computer Lab
Viale Monte Nero, 66
20135 Milano
Tel. 02/54.64.43.6

Computer Service
Via Genala, 19/B
26100 Cremona
Tel. 0372/43.58.61

MARCHE

Car
Via Bruno Buozzi, 18
60020 Ancona
Tel. 071/80.44.88

PIEMONTE

Grun A.R.
Via De Sanctis, 126/F
10142 Torino
Tel. 011/70.72.47.2

Computer Service Torino
Via Reiss Romoli, 122/11
10100 Torino
Tel. 011/22.02.66.6

PUGLIA

Viga
Via Domenico Morea, 42
70124 Bari
Tel. 080/41.37.66

SARDEGNA

Alacram Technologies
Via Livenza, 3
09123 Cagliari
Tel. 070/28.72.38

SICILIA

Cat Elettronica
Via Ravenna, 7/A
95100 Catania
Tel. 095/43.86.70

Co.As. Informatica
Via Raffaele Mondini, 3
90143 Palermo
Tel. 091/29.52.09

TOSCANA

G.L.V. Elettronica
Via Pietrasantina, 113
56100 Pisa
Tel. 050/56.20.35

Paolo Paolieri
Via Perfetti Ricasoli, 70
50100 Firenze
Tel. 055/43.61.72.0

TRENTINO ALTO ADIGE

Elecomp
Via Druso, 52/A
39100 Bolzano
Tel. 0471/42.12.8

UMBRIA

H.C.H.
Via Ruggero D'Andreotto, 31/A
06100 Perugia
Tel. 075/75.33.53

VENETO

Carpanese Elettronica Telecomunicazioni
Strada VII Martiri, 101
35100 Padova
Tel. 049/62.41.60

Sistel
Via Decorati al Valor Civile, 67
30100 Mestre
Tel. 041/93.53.32

Pesente Giovanni
Via Pitagora, 6
37100 Verona
Tel. 045/56.59.8

PRODOTTI SYSTEMS EDITORIALE

Software su cassetta

La voce III	L.12000
Raffaello	L.10000
Oroscopo	L.12000
Computer-Music	L.12000
Gestione familiare	L.12000
Banca dati	L.12000
Dichiarazione dei redditi (740/S)	L.16000
Matematica finanziaria	L.20000
Analisi di bilancio	L.20000
Arredare (richiede linguaggio Simon's Basic)	L.10000

Software su disco

Ms-Dos & Gw-Basic	L.25000
Ms-Dos & Gw-Basic (con prova di acquisto cassetta)	L.15000
La voce III	L.12000
Raffaello	L.10000
Oroscopo	L.12000
Computer-Music	L.12000
Gestione familiare	L.12000
Banca dati	L.12000
Dichiarazione dei redditi (740/S)	L.24000
Matematica finanziaria	L.20000
Analisi di bilancio	L.20000
Arredare (richiede linguaggio Simon's Basic)	L.20000
Graphic Expander C/128 in modo 80 colonne	L.27000
Linguaggio macchina + Routine grafiche	L.12000

Offerta "Commodore speciale L.M." + dischetto — L.16000

IN EDICOLA

systems

systems